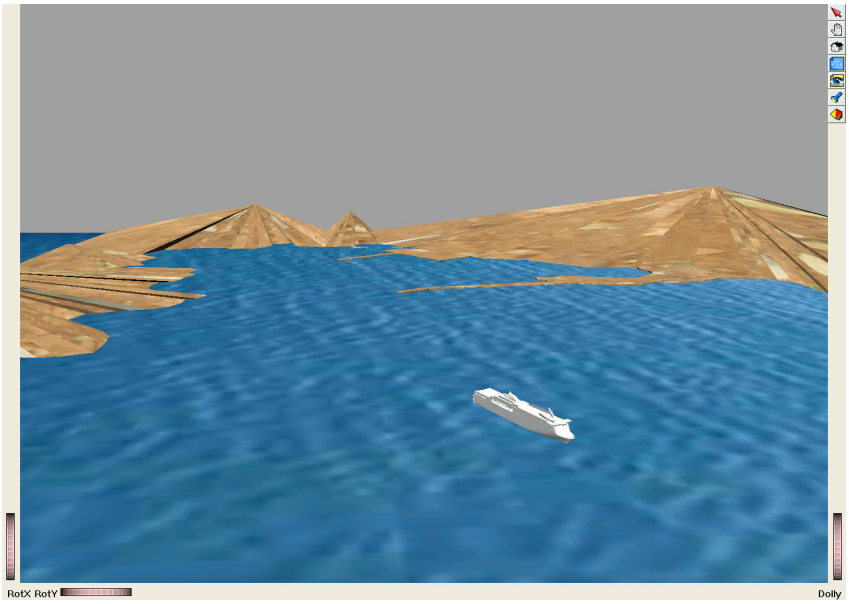
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΝΑΥΠΗΓΩΝ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΜΕΛΕΤΗΣ ΠΛΟΙΟΥ ΚΑΙ ΘΑΛΑΣΣΙΩΝ ΜΕΤΑΦΟΡΩΝ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

ΑΝΑΠΤΥΞΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΤΡΙΣΔΙΑΣΤΑΤΗΣ ΓΡΑΦΙΚΗΣ ΠΡΟΣΟΜΕΙΩΣΗΣ ΤΩΝ ΟΡΙΖΟΝΤΙΩΝ ΚΙΝΗΣΕΩΝ ΠΛΟΙΟΥ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΕΠΕΜΒΑΣΗΣ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ.



ΓΕΩΡΓΙΟΣ ΓΚΟΣΕΒΙΤΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ Κ. ΣΠΥΡΟΥ

ΑΘΗΝΑ ΜΑΙΟΣ 2007

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 4

ΓΕΩΜΕΤΡΙΚΗ ΑΠΕΙΚΟΝΙΣΗ

ΚΕΦΑΛΑΙΟ 1

1. ΕΙΣΑΓΩΓΗ.

Οι υπολογιστές και η δυνατότητα τρισδιάστατων γραφικών απεικονίσεων έχουν εισβάλει σε όλους τους τομείς της επιστήμης με πολλαπλές εφαρμογές. Χαρακτηριστικές περιπτώσεις εφαρμογής των τρισδιάστατων γραφικών απεικονίσεων στον κόσμο της επιστήμης είναι η χρησιμοποίησή τους στην μελέτη και επίλυση προβλημάτων μηχανικής. Πλέον, σε όλους τους κλάδους της μηχανικής οι υπολογιστές και οι γραφικές τους δυνατότητες αποτελούν ένα πολύτιμο "εργαλείο" στη διάθεση των επιστημόνων.

Στην ναυπηγική η χρήση των τρισδιάστατων γραφικών μπορεί να εφαρμοστεί στην μελέτη της φόρτωσης του πλοίου, στην μελέτη της μεταλλικής του κατασκευής και των δυνάμεων και τάσεων που ασκούνται σε αυτήν, στην υδροδυναμική μελέτη και γενικώς σε πολλά στάδια της μελέτης του πλοίου. Σημαντική εφαρμογή των τρισδιάστατων γραφικών απεικονίσεων είναι στη μελέτη της ελικτικής συμπεριφοράς του πλοίου και στην απόκρισή του σε περιβαλλοντικές επιδράσεις. Με τη χρήση των γραφικών μπορούν να εξαχθούν σημαντικά συμπεράσματα για την συμπεριφορά του πλοίου κατά την διάρκεια διαφόρων συνθηκών που είναι δυνατόν να συναντήσει κατά την λειτουργία του. Στον τομέα της μελέτης της ελικτικής συμπεριφοράς του πλοίου είναι σημαντικό να υπάρχει η δυνατότητα για παρατήρηση σε πραγματικό χρόνο (real time) και για αλληλεπίδραση του χρήστη με το γραφικό περιβάλλον.

Στόχος της παρούσης διπλωματικής εργασίας είναι η δημιουργία ενός περιβάλλοντος που θα επιτρέπει την τρισδιάστατη παρατήρηση των αποκρίσεων του πλοίου σε σχέση με την ελικτική του συμπεριφορά. Το

πρόγραμμα εκτός από την παρατήρηση των διάφορων κινήσεων του πλοίου θα επιτρέπει και την παρέμβαση του χρήστη σε πραγματικό χρόνο. (π.χ. μεταβάλλοντας την γωνία του πηδαλίου). Το τελευταίο στοιχείο είναι και το πιο σημαντικό αφού η δυνατότητα της αλληλεπίδρασης του χρήστη με το περιβάλλον εικονικής πραγματικότητας σε πραγματικό χρόνο είναι ιδιαίτερα χρήσιμο εργαλείο για την κατανόηση, μέσω της παρατήρησης και του πειραματισμού, της συμπεριφοράς του πλοίου κατά την διάρκεια διάφορων ελιγμών. Ο χρήστης αποκτά μια άμεση αίσθηση της ελικτικότητας του πλοίου.

Το πρόγραμμα θα αναπαριστά, σε πρώτη φάση, τις κινήσεις του πλοίου σε ήρεμο νερό απουσία κυματισμών και άλλων εξωτερικών παραμέτρων όπως η επίδραση ανέμου, ρευμάτων κ.ά. Δηλαδή έχουμε αναπαράσταση των κινήσεων του πλοίου σε τέσσερις βαθμούς ελευθερίας .Στην ουσία θα δίνεται η δυνατότητα να “κυβερνήσει” ο χρήστης το πλοίο σε ήρεμο νερό μεταβάλλοντας σε πραγματικό χρόνο την γωνία του πηδαλίου και τις στροφές της έλικας. Μπορούν να γίνουν διάφορες δοκιμές της ελικτικής συμπεριφοράς του πλοίου (δοκιμές zig-zag, κύκλοι στροφής κ.α.). Επίσης θ’ αναπτυχθούν τρισδιάστατες απεικονίσεις των λιμανιών της Σύρου και της Τήνου ώστε να μπορούν να γίνουν δοκιμές προσέγγισής τους. Όλα τα παραπάνω θα εφαρμοστούν στο επιβατηγό/οχηματαγωγό (Ro-Ro) Αλκυών το οποίο σχεδιάστηκε πριν από λίγα χρόνια σε συνεργασία του Πολυτεχνείου με τα Ναυπηγεία Ελευσίνας. Το τελικό αποτέλεσμα λοιπόν είναι ένα πρόγραμμα προσομοίωσης (simulation) για το επιβατηγό/οχηματαγωγό Αλκυών σε ήρεμο νερό.

Για την δημιουργία του παραπάνω προγράμματος γίνεται χρήση της γλώσσας C++ και των “εξωτερικών βιβλιοθηκών” Coin3d (Open Inventor) και IMSL.C. Η βιβλιοθήκη Coin3d χρησιμοποιείται για την απεικόνιση και χρησιμοποίηση των τρισδιάστατων γραφικών σε πραγματικό χρόνο και η βιβλιοθήκη IMSL.C για την επίλυση του συστήματος των διαφορικών εξισώσεων που διέπουν την κίνηση του πλοίου. Συνεπώς, το πρόγραμμα μπορούμε να πούμε ότι αποτελείται από δύο μέρη που αλληλοεπιδρούν

μεταξύ τους κατά την διάρκεια του χρόνου. Το ένα μέρος αφορά την μαθηματική πτυχή του προβλήματος αναλύοντας τις δυνάμεις και τις ροπές που ασκούνται στο πλοίο και επιλύοντας το σύστημα των διαφορικών εξισώσεων στο οποίο υπακούν οι κινήσεις του και το δεύτερο μέρος χρησιμοποιεί τα αποτελέσματα αυτά για να απεικονίσει τρισδιάστατα το πλοίο. Η αλληλεπίδραση των δύο αυτών σκελών βρίσκεται στο γεγονός ότι όλα γίνονται σε πραγματικό χρόνο. Έτσι η διαδικασία που ακολουθείται είναι:

απεικόνιση -> επίλυση συστήματος -> απεικόνιση -> επίλυση συστήματος -> κτλ., κάθε δt second που έχουμε ορίσει.

Στο παρόν σύγγραμμα θα αναφερθούμε αρχικά στο μαθηματικό μοντέλο που διέπει τις κινήσεις του πλοίου. Στην συνέχεια θα παρουσιάσουμε τον κώδικα που εκφράζει το μαθηματικό μοντέλο και την επίλυση του συστήματος των διαφορικών εξισώσεων .Στο επόμενο στάδιο αναλύονται οι βασικές εφαρμογές της βιβλιοθήκης Coin3d που επιτρέπουν την τρισδιάστατη γραφική απεικόνιση. Γίνεται επεξήγηση του πώς εφαρμόζονται τα αποτελέσματα του μαθηματικού μοντέλου στο πλοίο σε πραγματικό χρόνο και πως αποκτάται η δυνατότητα της αλληλεπίδρασης με τον χρήστη για την δημιουργία του τελικού προσομοιωτή. Τέλος γίνεται αναφορά στη δυνατότητα επέκτασης του προγράμματος ώστε να περιλαμβάνει και τις κινήσεις σε κυματισμούς με προσθήκη των κινήσεων heave και pitch, την εισαγωγή και άλλων πλοίων και στις πολλαπλές εφαρμογές που προσφέρει η βιβλιοθήκη Coin3d για τρισδιάστατες γραφικές απεικονίσεις σε πραγματικό χρόνο.

ΚΕΦΑΛΑΙΟ 2

ΕΙΣΑΓΩΓΗ ΣΤΗ ΘΕΩΡΙΑ ΤΩΝ ΕΛΙΚΤΙΚΩΝ ΙΚΑΝΟΤΗΤΩΝ

2.1 ΒΑΣΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΚΙΝΗΣΗΣ

Για την περιγραφή ενός επιπλέοντας σώματος χρησιμοποιούνται οι εξισώσεις κίνησης που ορίζονται από την δυναμική στερεού σώματος σύμφωνα με τον δεύτερο νόμο του Νεύτωνα. Οι ροπές και δυνάμεις που ασκούνται σε ένα σώμα είναι ίσες με την μεταβολή τής στροφορμής και της ορμής αντίστοιχα.

$$F = \frac{d(m\vec{U})}{dt} \quad (1) \qquad \& \qquad M = \frac{d(I\vec{\Omega})}{dt} \quad (2)$$

όπου,

m : η μάζα του σώματος

$\vec{U}$: η ταχύτητα του σώματος

I : η ροπή αδράνειας του σώματος περί τον άξονα περιστροφής

$\vec{\Omega}$: η γωνιακή ταχύτητα του σώματος.

Οι παραπάνω εξισώσεις χρησιμοποιούνται για την περιγραφή της κίνησης του πλοίου στους έξι βαθμούς ελευθερίας. Με τον όρο έξι βαθμούς ελευθερίας εννοούμε τις έξι δυνατές κινήσεις που μπορεί να εκτελέσει το πλοίο στον χώρο. Οι κινήσεις αυτές βρίσκονται συγκεντρωμένες στον παρακάτω πίνακα (Πίνακας 1) :

ΠΙΝΑΚΑΣ 1

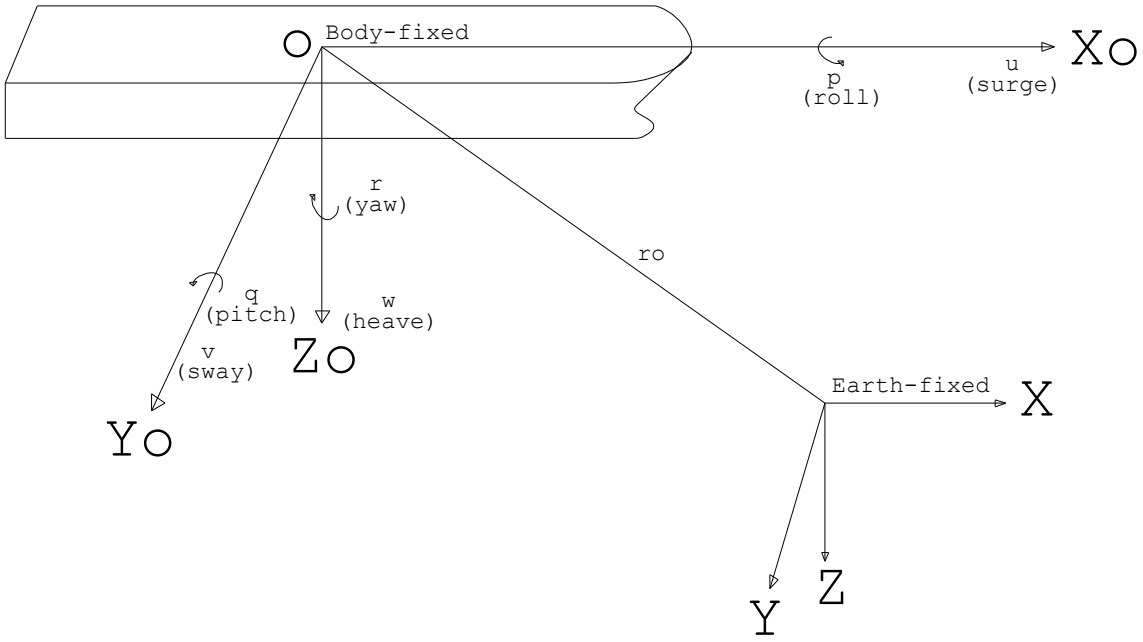
Β.Ελευθερίας	Περιγραφή	Δυνάμεις-Ροπές	Γραμμικες και Γωνιακες ταχύτητες	Θέσεις και Γωνίες
1	Κίνηση κατα τον x άξονα	X	u	x
2	Κίνηση κατα τον y άξονα	Y	v	y
3	Κίνηση κατα τον z άξονα	Z	w	z
4	Περιστροφή κατα τον x άξονα	K	p	φ
5	Περιστροφή κατα τον y άξονα	M	q	θ
6	Περιστροφή κατα τον z άξονα	N	r	ψ

Πίνακας 1.Κινήσεις στους έξι βαθμούς ελευθερίας

Για να ορίσουμε την κίνηση του πλοίου στους έξι βαθμούς ελευθερίας είναι απαραίτητο να οριστούν δύο συστήματα συντεταγμένων. Το πρώτο έχει το κέντρο του πάνω στην μέση τομή του πλοίου και συνεπώς είναι κινούμενο. Είναι καρτεσιανό και οι άξονες του ορίζονται ως Xo, Yo, Zo. Οι κινήσεις του πλοίου ορίζονται ως προς το σωματοπαγές σύστημα συντεταγμένων. Κίνηση και περιστροφή κατά τον άξονα x ορίζουν τις κινήσεις surge και roll αντιστοίχως. Ομοίως, η κίνηση και η περιστροφή κατά τον άξονα y ορίζουν τις κινήσεις sway και pitch αντιστοίχως. Τέλος, κίνηση και περιστροφή κατά τον άξονα z ορίζουν τις κινήσεις heave και roll αντιστοίχως.

Το δεύτερο σύστημα συντεταγμένων βρίσκεται πάνω στην γη και είναι ακίνητο, καρτεσιανό με άξονες X, Y, Z. Το σύστημα αυτό είναι απαραίτητο για τον παρατηρητή που βρίσκεται έκτος του πλοίου σε κάποιο σημείο της στεριάς. Το σημαντικό πλεονέκτημα αυτού του συστήματος είναι ότι μας δίνει την δυνατότητα να παρατηρήσουμε την τροχιά που διαγράφει το πλοίο πάνω στο επίπεδο της θάλασσας καθώς και τις υπόλοιπες κινήσεις του φυσικά. Συνήθως ως αρχή του συστήματος αυτού λαμβάνεται ένα σημείο το οποίο καθ' ύψος βρίσκεται στο επίπεδο της θάλασσας.

Στο παρακάτω σχήμα παρουσιάζονται οι ορισμοί των συστημάτων που αναφέραμε (Σχήμα 1)



Σχήμα 1. Συστήματα Συντεταγμένων

2.2 ΛΕΠΤΟΜΕΡΗΣ ΜΟΡΦΗ ΤΩΝ ΕΞΙΣΩΣΕΩΝ ΚΙΝΗΣΗΣ

Σύμφωνα με την δυναμική στερεού σώματος οι εξισώσεις κίνησης του πλοίου στους έξι βαθμούς ελευθερίας όπως ορίζονται απ' το σωματοπαγές σύστημα συντεταγμένων έχουν την παρακάτω μορφή:

Surge:

$$m(\dot{u} + qw - rv - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(\dot{q} + pr)) = X_H + X_P + X_R + X_0 \qquad (3)$$

Sway:

$$m(\dot{v} + ru - pw - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})) = Y_H + Y_P + Y_R + Y_0 \qquad (4)$$

Heave:

$$m(\dot{w} + pv - qu - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + y_G(rq + \dot{p}) = Z_H + Z_P + Z_R + Z_0 \qquad (5)$$

Roll:

$$I_x\dot{p} + (I_z - I_y)qr + m[y_G(\dot{w} + pv - qu) - z_G(\dot{v} + ru - pw)] = K_H + H_P + K_R + K_0 \qquad (6)$$

Pitch :

$$I_y\dot{q} + (I_x - I_z)rp + m[z_G(\dot{u} + qw - rv) - x_G(\dot{w} + pv - qu)] = M_H + M_P + M_R + M_0 \qquad (7)$$

Yaw:

$$I_z\dot{r} + (I_y - I_x)pq + m[x_G(\dot{v} + ru - pw) - y_G(\dot{u} + qw - rv)] = N_H + N_P + N_R + N_0 \qquad (8)$$

Όπου,

m : η μάζα του πλοίου

u, v, w : οι γραμμικές ταχύτητες (Σχήμα 1)

$\dot{u}, \dot{v}, \dot{w}$: οι γραμμικές επιταχύνσεις

x_G, y_G, z_G : η απόσταση του κέντρου βάρους του πλοίου από την μέση τομή

Ix, Iy, Iz : οι ροπές αδράνειας ως προς το σωματοπαγές σύστημα αξόνων του πλοίου.

p, q, r : οι γωνιακές ταχύτητες (Σχήμα 1)

$\dot{p}, \dot{q}, \dot{r}$: οι γωνιακές επιταχύνσεις

X, Y, Z, K, M, N : δυνάμεις και ροπές λόγω υδροδυναμικών επιδράσεων στην γάστρα (δείκτης H) , στην έλικα (δείκτης P), στο πηδάλιο (δείκτης R) και λόγω εξωτερικών παραγόντων όπως η επίδραση ανέμου, κλιμάτων κτλ. (δείκτης O).

Όπως αναφέρθηκε και προηγουμένως, στην παρούσα διπλωματική εργασία θα προσομοιάσουμε τις κινήσεις του πλοίου σε ήρεμο νερό απουσία κυματισμών και άλλων διαταραχών. Συνεπώς, πιο ενδιαφέρουσες (στην παρούσα φάση) κρίνονται οι κινήσεις κατά τις διευθύνσεις surge και sway και οι περιστροφές κατά τις διευθύνσεις yaw και roll. Παραλείποντας τις εξισώσεις που περιγράφουν τις κινήσεις heave και pitch, μηδενίζοντας τους όρους p και q και με κέντρο βάρους του πλοίου ένα σημείο (x_G,0, z_G) οι εξισώσεις που περιγράφουν τις υπόλοιπες κινήσεις του πλοίου είναι:

Surge:

$$m(\dot{u} - rv - x_G r^2 + z_G pr) = X_H + X_P + X_R + X_0 \qquad (9)$$

Sway:

$$m(\dot{v} + ru - z_G \dot{p} + x_G \dot{r}) = Y_H + Y_P + Y_R + Y_0 \qquad (10)$$

Yaw:

$$I_z \dot{r} + m x_G (\dot{v} + ru) = N_H + N_P + N_R + N_0 \qquad (11)$$

Roll:

$$I_x \dot{p} - m z_G (\dot{v} + ru) = K_H + K_P + K_R + K_0 \qquad (12)$$

2.3 ΑΝΑΛΥΣΗ ΥΔΡΟΔΥΝΑΜΙΚΩΝ ΔΥΝΑΜΕΩΝ

Οι υδροδυναμικές δυνάμεις που ασκούνται επί του πλοίου κατά τα διάφορα στάδια των ελιγμών αφορούν την γάστρα ,το πηδάλιο ,την έλικα και τις αλληλεπιδράσεις τους. Οι αλληλεπιδράσεις έχουν ιδιαίτερη σημασία στην περίπτωση του πηδαλίου το οποίο δέχεται την επιταχυνόμενη ροή λόγω της έλικας.

Το πρόβλημα του υπολογισμού των δυνάμεων αυτών είναι ιδιαίτερα πολύπλοκο (λόγω της σύνθετης φύσης αυτών των δυνάμεων) και έως

σήμερα δεν υπάρχει ένα μαθηματικό μοντέλο με το οποίο να μπορούμε να υπολογίσουμε με αυστηρή ακρίβεια αυτές τις φορτίσεις.
Προφανώς, η γνώση αυτών των φορτίσεων είναι απαραίτητη εάν θέλουμε να προβλέψουμε την συμπεριφορά ενός πλοίου. Γι αυτόν τον λόγο έχουν αναπτυχθεί μεθοδολογίες βασισμένες σε πειράματα και θεωρία που με απλές εμπειρικές παραδοχές δίνουν αποτελέσματα για τις τιμές των φορτίσεων που ασκούνται επί του πλοίου.

Δύο είναι οι βασικές υποθέσεις – παραδοχές:

- Οτι η κίνηση ελιγμού του πλοίου είναι αργή (συνήθως αληθές για τα περισσότερα εμπορικά πλοία)

- Οτι οι δυνάμεις και οι ροπές κατά μια χρονική στιγμή t δεν εξαρτώνται από την κατάστασή τους την προηγούμενη χρονική στιγμή t – dt. Η υπόθεση αυτή ουσιαστικά συναρτάται με την πρώτη.

Συνεπώς, η κλασική παραδοχή που γίνεται είναι ότι οι δυνάμεις που ασκούνται στο πλοίο (γάστρα, έλικα, πηδάλιο) μπορούν να εκφραστούν σαν συνάρτηση όλων των παραμέτρων της κίνησης, δηλαδή:

$$X = f_1(u, v, r, \dot{u}, \dot{v}, \dot{r}) \qquad \text{Surge} \qquad (13)$$

$$Y = f_2(u, v, r, \dot{u}, \dot{v}, \dot{r}) \qquad \text{Sway} \qquad (14)$$

$$N = f_3(u, v, r, \dot{u}, \dot{v}, \dot{r}) \qquad \text{Yaw} \qquad (15)$$

$$K = f_4(u, v, r, \dot{u}, \dot{v}, \dot{r}) \qquad \text{Roll} \qquad (16)$$

Ως γνωστόν,οι αναλυτικές συναρτήσεις μπορούν να προσεγγιστούν με την σειρά Taylor και να προσδιοριστούν οι τιμές τους σε μία θέση Χο σύμφωνα με την σχέση:

$$f(x) = f(x_0) + \frac{\delta x}{1!}\frac{df(x)}{dx}_{x=x_0} + \frac{\delta x^2}{2!}\frac{df^2(x)}{dx^2}_{x=x_0} + \ldots \qquad (17)$$

Στην περίπτωση που έχουμε συναρτήσεις πολλών μεταβλητών εφαρμόζουμε παρόμοια διαδικασία για τις υδροδυναμικές δυνάμεις και παίρνουμε τελικά:

$$X = f_1(u_0, v_o, r_o, \dot{u}_0, \dot{v}_o, \dot{r}_o) + \delta u \frac{\partial X}{\partial u} + \ldots + \delta\ddot{r}\frac{\partial X}{\partial \dot{r}} \qquad \text{Surge} \qquad (18)$$

$$Y = f_2(u_0, v_o, r_o, \dot{u}_0, \dot{v}_o, \dot{r}_o) + \delta u \frac{\partial Y}{\partial u} + \ldots + \delta\ddot{r}\frac{\partial Y}{\partial \dot{r}} \qquad \text{Sway} \qquad (19)$$

$$N = f_3(u_0, v_o, r_o, \dot{u}_0, \dot{v}_o, \dot{r}_o) + \delta u \frac{\partial N}{\partial u} + \ldots + \delta\ddot{r}\frac{\partial N}{\partial \dot{r}} \qquad \text{Yaw} \qquad (20)$$

$$K = f_4(u_0, v_o, r_o, \dot{u}_0, \dot{v}_o, \dot{r}_o) + \delta u \frac{\partial K}{\partial u} + \ldots + \delta\ddot{r}\frac{\partial K}{\partial \dot{r}} \qquad \text{Roll} \qquad (21)$$

Με ορισμένες βασικές παραδοχές πολλοί όροι των παραπάνω απειροσειρών μηδενίζονται. Συνήθως κρατιούνται μόνο οι γραμμικοί όροι α’ τάξης και παραλείπονται οι υπόλοιποι απλουστεύοντας έτσι σημαντικά τις σχέσεις αν

και οι απόψεις διίστανται για το ποιοι ακριβώς όροι πρέπει να παραληφθούν.

Μία αρκετά καλή αντιμετώπιση του προβλήματος είναι η χρήση των υδροδυναμικών παραγώγων για τον προσδιορισμό των υδροδυναμικών δυνάμεων όπως αυτές παρουσιάζονται στις εργασίες των Wagner-Smitt,Inoue και στην παλινδρομική ανάλυση που εκτέλεσε ο Clarke για τους τύπους των προηγουμένων. Τελικώς , έχουμε τις παρακάτω εκφράσεις για τις υδροδυναμικές παραγώγους κατά Clarke: [1]

$$Y_{\dot{v}}' = -\pi(\frac{T}{L})^2[1 + 0.16C_b\frac{B}{T} - 5.1(\frac{B}{L})^2] \qquad Y_{\dot{r}}' = -\pi(\frac{T}{L})^2[0.67\frac{B}{L} - 0.0033(\frac{B}{T})^2]$$

$$N_{\dot{v}}' = -\pi(\frac{T}{L})^2[1.1\frac{B}{L} - 0.041\frac{B}{T}] \qquad N_{\dot{r}}' = -\pi(\frac{T}{L})^2[\frac{1}{12} + 0.017C_b\frac{B}{T} - 0.33\frac{B}{L}]$$

$$Y_v' = -\pi(\frac{T}{L})^2[1 + 0.4C_b\frac{B}{T}] \qquad Y_r' = -\pi(\frac{T}{L})^2[-0.5 + 2.2\frac{B}{L} + 0.08\frac{B}{T}]$$

$$N_v' = -\pi(\frac{T}{L})^2[0.5 + 2.4\frac{T}{L}] \qquad N_r' = -\pi(\frac{T}{L})^2[0.25 + 0.039\frac{B}{T} - 0.056\frac{B}{L}]$$

2.4 ΔΥΝΑΜΕΙΣ ΚΑΙ ΡΟΠΕΣ ΠΟΥ ΕΝΕΡΓΟΥΝ ΕΠΙ ΤΗΣ ΓΑΣΤΡΑΣ

Στο παρόν εδάφιο θα αναλύσουμε τους όρους που βρίσκονται στο δεξί μέλος των σχέσεων (9), (10), (11), (12) που αφορούν τις δυνάμεις και τις ροπές που ασκούνται στο πλοίο.

Οι ασκούμενες δυνάμεις και ροπές στην γάστρα του πλοίου δίνονται από τις παρακάτω σχέσεις:

$$X_H = X_{\dot{u}}\dot{u} - Y_v vr - \frac{u}{|u|}Y_r r^2 + X_{vr}vr + X(u) \qquad \text{(surge)} \qquad (22)$$

$$Y_H = Y_{\dot{v}}\dot{v} + Y_{\dot{r}}\dot{r} + Y_v vU + \frac{u}{|u|}Y_r rU + Y_{vv}v|v| + Y_{vr}v|r| + \frac{u}{|u|}Y_{rr}r|r| \qquad \text{(sway)} \qquad (23)$$

$$N_H = N_{\dot{r}}\dot{r} + N_{\dot{v}}\dot{v} + N_r rU + \frac{u}{|u|}N_v vU + N_{rr}r|r| + \frac{u}{|u|}N_{rrv}\frac{rrv}{U} + N_{vvr}\frac{vvr}{U} + \frac{u}{|u|}N_{\phi\phi}\phi U^2 + \frac{u}{|u|}N_{v\phi}v|\phi|U + N_{r\phi}r|\phi|U \qquad \text{(yaw)} \qquad (24)$$

$$K_H = K_{\dot{p}}\dot{p} + K_p p + K_\phi \sin\phi - z_Y Y_H \qquad \text{(roll)} \qquad (25)$$

Ο πολλαπλασιασμός με τον όρο $\frac{u}{|u|}$,που ισούται με 1 ή -1, εκτελείται για να ληφθεί υπ' όψη πρόσω ή όπισθεν κίνηση του πλοίου.

$U = \sqrt{u^2 + v^2}$, είναι η συνισταμένη γραμμική ταχύτητα του πλοίου

X(u): η αντίσταση του πλοίου κατά την πρόσω κίνηση.

Zy : είναι το κέντρο δράσης της υδροδυναμικής δύναμης Y_H που εκτιμάται ότι βρίσκεται στο 40% του βυθίσματος κάτω από την ίσαλο.

2.5 ΔΥΝΑΜΕΙΣ ΚΑΙ ΡΟΠΕΣ ΣΤΗΝ ΕΛΙΚΑ

Η βασική δύναμη που ασκείται στην έλικα του πλοίου είναι προφανώς κατά την διεύθυνση της κίνησης surge και ισούται με:

$$Xp = (1 - t)T \qquad (26)$$

Όπου,

t: το ποσοστό μείωσης ώσης

T: η ώση που παράγει η έλικα

Όσον αφορά τις κινήσεις στις υπόλοιπες διευθύνσεις έχουμε:
Κατά την περιστροφή roll δεν ασκείται ροπή στην έλικα και συνεπώς λαμβάνουμε Kp = 0.

Η πλευρική δύναμη sway και η ροπή κατά yaw έχουν μικρή συνεισφορά κατά την πρόσω κίνηση του πλοίου και συνεπώς μηδενίζονται. Στην περίπτωση της πίσω κίνησης αποκτούν μεγαλύτερες τιμές και λαμβάνονται υπ'όψη στο μαθηματικό μοντέλο.

2.6 ΔΥΝΑΜΕΙΣ ΚΑΙ ΡΟΠΕΣ ΣΤΟ ΠΗΔΑΛΙΟ

Είναι γνωστό ότι το πηδάλιο αποτελεί πολύ σημαντικό παράγοντα στην ελικτική συμπεριφορά του πλοίου. Φυσικά, το πηδάλιο δεν στρίβει από μόνο του το πλοίο αλλά τοποθετεί την γάστρα σε θέση ώστε να την στρίψει η ροή του περιρρέοντως νερού. Μια καλή προσέγγιση των δυνάμεων και ροπών που ασκούνται στο πηδάλιο δίνονται από τις παρακάτω σχέσεις:

$X_R = (1+t_R)F_n \sin\delta$ Surge (27)

$Y_R = (1+a_H)F_n \cos\delta$ Sway (28)

$N_R = (1+a_H)x_R F_N \cos\delta$ Yaw (29)

$K_R = -(1+a_H)z_R F_N \cos\delta$ Roll (30)

Όπου,

δ: η γωνία εκτροπής του πηδαλίου

x_R, z_R: η διαμήκης και κατακόρυφη απόσταση από το κέντρο του σωματοπαγούς συστήματος συντεταγμένων (μέση τομή) μέχρι το σημείο εφαρμογής της ανωστικής δύναμης του πηδαλίου.

t_R: το ποσοστό μείωσης ώσης.

a_H: ο συντελεστής αλληλεπίδρασης πηδαλίου – γάστρας

F_N: κάθετη δύναμη που ασκείται στο πηδάλιο και ισούται με:

$$F_N = \frac{1}{2}\rho A_R U_R{}^2 f(\Lambda)\sin a_R$$ όπου,

ρ: το ειδικό βάρος του νερού

A_R: η επιφάνεια του πηδαλίου

U_R: μέση ταχύτητα της ροής γύρο από το πηδάλιο

a_R: η γωνία πρόσπτωσης της ροής στο πηδάλιο

f(Λ): συντελεστής κάθετης δύναμης του πηδαλίου σε ελεύθερη ροή χωρίς γάστρα που με ικανοποιητική ακρίβεια προσεγγίζεται από την σχέση

$f(\Lambda) = 6.13\lambda/\lambda + 2.25$ όπου,

λ: ο λόγος επιμήκους του πηδαλίου.

Το μαθηματικό μοντέλο που χρησιμοποιείται στην παρούσα διπλωματική εργασία βασίζεται στην θεωρία που αναπτύχθηκε παραπάνω. Στο επόμενο

κεφάλαιο θα περιγράψουμε αναλυτικά τον κώδικα που χρησιμοποιεί το μαθηματικό μοντέλο και εκτελεί τους υπολογισμούς

ΚΕΦΑΛΑΙΟ 3

ΑΝΑΠΤΥΞΗ ΑΛΛΗΛΕΠΙΔΡΑΣΤΙΚΟΥ ΚΩΔΙΚΑ ΠΡΟΣΟΜΕΙΩΣΗΣ

3.1 ΓΕΝΙΚΑ

Η παρούσα διπλωματική εργασία διεκπεραιώθηκε με την χρησιμοποίηση του Microsoft Visual Studio v 6.0. Πρόκειται για ένα περιβάλλον εργασίας όπου ο χρήστης μπορεί να γράψει τον κώδικά του, να τον κάνει compile και να τον εκτελέσει (execute).Το περιβάλλον αυτό είναι σε μορφή windows και είναι πιο φιλικό στον προγραμματιστή από τους υπόλοιπους compiler που δουλεύουν μέσω του command prompt (MS DOS).

Η γλώσσα προγραμματισμού που επιλέχθηκε είναι η C++ διότι σε αυτήν την γλώσσα είναι γραμμένη η βιβλιοθήκη Coin3d που περιλαμβάνει τις απαραίτητες συναρτήσεις και λειτουργίες για την δημιουργία του περιβάλλοντος προσομείωσης. Η βιβλιοθήκη Coin3d είναι γραμμένη "πάνω" στην βιβλιοθήκη OpenGl. Η βιβλιοθήκη OpenGL είναι μία εκ των βασικών βιβλιοθηκών (η άλλη είναι η DirectX) που περιλαμβάνουν συναρτήσεις και λειτουργίες για την αναπαράσταση τρισδιάστατων γραφικών. Η βιβλιοθήκη OpenGL επικοινωνεί άμεσα με την κάρτα γραφικών του υπολογιστή και ορίζει σε αυτήν τι πρόκειται να ζωγραφιστεί. Η χρησιμοποίηση όμως της βιβλιοθήκης OpenGL απαιτεί γνώσεις προγραμματισμού επαγγελματικού επιπέδου και είναι ιδιαίτερα δύσκολο εγχείρημα. Για τον λόγο αυτό υπάρχουν βιβλιοθήκες όπως η Coin3d που ουσιαστικά αποτελούν ένα πιο εύχρηστο και πρακτικό τρόπο χρησιμοποίησης της βιβλιοθήκης OpenGL.

Επίσης, χρησιμοποιήθηκε η βιβλιοθήκη IMSL.C η οποία περιλαμβάνει μαθηματικές και στατιστικές συναρτήσεις . Η βιβλιοθήκη IMSL.C διαθέτει

το απαραίτητο λογισμικό που απαιτεί η επίλυση σύνθετων προβλημάτων αριθμητικής ανάλυσης. Ουσιαστικά, απαλλάσσει τον προγραμματιστή από το να συντάξει τον δικό του κωδικά αφού περιέχει γραμμένους μαθηματικούς και στατιστικούς αλγορίθμους που καλούνται μέσω της γλώσσας C++.

Από προηγούμενη διπλωματική εργασία προϋπήρχε σε γλώσσα Fortran ένας κώδικας που υπολόγιζε τις οριζόντιες κινήσεις του πλοίου απουσία κυματισμών. Ο κώδικας αυτός ανέλυε τις δυνάμεις κατά τις διευθύνσεις surge και sway και τις ροπές κατά τις διευθύνσεις yaw και roll. Η επίλυση του συστήματος των διαφορικών εξισώσεων που διέπουν τις κινήσεις του πλοίου γίνονταν με χρήση της συνάρτησης DivPRK της βιβλιοθήκης IMSL (έκδοση για Fortran). Η συνάρτηση DivPRK αποτελεί την αριθμητική επαναληπτική μέθοδο Runge – Kutta για την επίλυση του συστήματος των διαφορικών εξισώσεων.

Στην παρούσα διπλωματική εργασία ο παραπάνω κώδικας μεταφράστηκε σε γλώσσα C++, τροποποιήθηκε η δομή του ώστε να ικανοποιεί τις απαιτήσεις του συνολικού προγράμματος και με χρήση της βιβλιοθήκης IMSL.C (έκδοση για C,C++) αποτελεί το μαθηματικό, υπολογιστικό μέρος του προσομοιωτή.

3.2 Η ΔΟΜΗ ΤΟΥ ΚΩΔΙΚΑ

Όπως αναφέραμε στην εισαγωγή, ο σκοπός της συγκεκριμένης προσπάθειας είναι να δημιουργηθεί ένας κώδικάς στην γλώσσα C++, χρησιμοποιώντας την βιβλιοθήκη Open Inventor η όποια αναπαριστά 3D γραφικά με χρήση των βιβλιοθηκών OpenGL και δίνει την δυνατότητα να τα χειριστούμε. Ο κώδικας αυτός θα πρέπει να μας δίνει την δυνατότητα όχι μόνο να παρακολουθούμε τις κινήσεις του πλοίου έχοντας δώσει κάποιες αρχικές συνθήκες στο πρόβλημα αλλά να μπορούμε να επεμβαίνουμε σε

πραγματικό χρόνο και να μεταβάλλουμε κάποιες τιμές συγκεκριμένων μεγεθών (π.χ. την γωνία του πηδαλίου) ώστε να υπάρχει πλήρης ελευθέρια κινήσεων σε ένα χρήστη να χειριστεί,στην ουσία να κυβερνήσει ,το δοθέν πλοίο.

Με την δυνατότητα της επέμβασης στο σύστημα μας σε πραγματικό χρόνο μπορούμε να προσομοιάσουμε συνθήκες πού είναι πιθανό να συναντήσει το πλοίο κατά την λειτουργία του και να έχουμε μια πρώτη εκτίμηση της απόκρισής του σε διάφορες απότομες ή μη μεταβολές της γωνίας του πηδαλίου και των στροφών της έλικας.

Για να επιτευχθεί ο παραπάνω στόχος το πρόγραμμα θα πρέπει να εκτελεί δύο διαδικασίες συγχρόνως ώστε να έχουμε το επιθυμητό αποτέλεσμα της αναπαράστασης σε πραγματικό χρόνο και την δυνατότητα επιβολής στο σύστημά μας. Η πρώτη διαδικασία βασίζεται στο μαθηματικό μοντέλο. Υπολογίζει τις δυνάμεις και τις ροπές που ασκούνται στο πλοίο. Η δεύτερη διαδικασία λαμβάνει τις τιμές της πρώτης και αναλαμβάνει την επίλυση του συστήματος των διαφορικών εξισώσεων που διέπουν την κίνηση του πλοίου και την τρισδιάστατη γραφική απεικόνιση. Επίσης αναλαμβάνει και τον παράγοντα χρόνο. Έτσι κάθε δt second που έχουμε ορίσει πρέπει να επιλύσει εκ νέου το σύστημα των διαφορικών εξισώσεων και στην συνέχεια να εφαρμόσει τα αποτελέσματα της επίλυσης καταλλήλως. Τέλος κατά την διάρκεια της δεύτερης διαδικασίας ορίζεται και η παρέμβαση του χρήστη στο εικονικό περιβάλλον.

Πιο αναλυτικά, η πρώτη διαδικασία αποτελείται από δύο αρχεία, το shipspecs.h και το functions.cpp. Η δεύτερη διαδικασία αποτελείται από το αρχείο simulation.cpp.

Το αρχείο Shipspecs.h αποτελεί header file του κώδικα και περιλαμβάνει τις τιμές των υδροδυναμικών παραγώγων, των συντελεστών ομόρου και μείωσης ώσης, τα γεωμετρικά στοιχεία του πλοίου και γενικότερα όλους

τους σταθερούς όρους που χρησιμοποιούνται στην επίλυση του μαθηματικού μοντέλου.

Το αρχείο Functionc.cpp περιλαμβάνει τις συναρτήσεις που χρησιμοποιούν τους σταθερούς όρους που βρίσκονται στο header file shipspecs.h και υπολογίζουν τις διάφορες δυνάμεις και ροπές που ασκούνται στο πλοίο. Είναι το καθαρά υπολογιστικό σκέλος του προγράμματος.

Το αρχείο Simulation.cpp περιλαμβάνει τον κώδικα που εκτελείται. Ο κώδικας λύνει το σύστημα των διαφορικών εξισώσεων σε πραγματικό χρόνο (κάθε 0.1 sec) χρησιμοποιώντας τα αποτελέσματα του Functions.cpp και την βιβλιοθήκη IMSL.C με τελικό αποτέλεσμα να "ζωγραφίζει" σε ένα παράθυρο windows το πλοίο ,την θάλασσα όλα τα γραφικά και τις κινήσεις στις οποίες υπακούν σύμφωνα με την λύση του συστήματος των διαφορικών εξισώσεων. Επίσης σε αυτό το σκέλος ορίζονται και οι αλληλεπιδράσεις που θα έχει ο χρήστης με το εικονικό περιβάλλον.

Το πρώτο και δεύτερο σκέλος του κώδικα, δηλαδή το header file shipspecs.h και το αρχείο functions.cpp είναι γραμμένα με ¨απλή¨ γλώσσα C++ δηλαδή χωρίς την χρησιμοποίηση επιπλέον συναρτήσεων, μεθόδων εξωτερικών βιβλιοθηκών. Το τρίτο σκέλος όπου βρίσκεται η main (η "συνάρτηση" που εκτελείται) χρησιμοποιεί την βιβλιοθήκη Open Inventor και τις συναρτήσεις που περιλαμβάνονται σε αυτήν για την αναπαράσταση και τον χειρισμό των γραφικών σε πραγματικό χρόνο. Σε αυτό το σκέλος χρησιμοποιείται και η βιβλιοθήκη IMSL.C με την ρουτίνα imsl_d_ode_runge_kutta για την επίλυση του συστήματος των διαφορικών εξισώσεων.

Το μοντέλο ελικτικότητας που χρησιμοποιείται βασίζεται στο θεωρητικό μοντέλο που αναπτύχθηκε στο κεφάλαιο ΙΙ. Πρόκειται για ένα κώδικα σε γλώσσα C++ ο οποίος δέχεται ως εισόδους αρχικές συνθήκες που αφορούν τις εξισώσεις κίνησης και υπολογίζει τις δυνάμεις και ροπές που ασκούνται στο πλοίο και διαμορφώνει το σύστημα των διαφορικών εξισώσεων.

3.3 ΑΝΑΛΥΣΗ ΤΩΝ ΑΡΧΕΙΩΝ

A) Header file Shipspecs.h

Αρχικά δημιουργείται το αρχείο shipspecs.h το οποίο περιέχει τις τιμές των υδροδυναμικών παραγώγων,των συντελεστών ομόρου και μείωσης ώσης, τα γεωμετρικά στοιχεία του πλοίου και γενικότερα όλους τους σταθερούς όρους που χρησιμοποιούνται στην επίλυση του μοντέλου ελικτικότητας. Αναλυτικά οι όροι αυτοί είναι:

ΠΙΝΑΚΑΣ 2

ΠΑΡΑΜΕΤΡΟΣ	**ΠΕΡΙΓΡΑΦΗ**	
YY_{vdot}	Υδροδυναμική παράγωγος	$\frac{\partial Y}{\partial \dot{v}}$
YYr_{dot}	Υδροδυναμική παράγωγος	$\frac{\partial Y}{\partial \dot{r}}$
YX_{vr}	Υδροδυναμική παράγωγος	$\frac{\partial^2 X}{\partial v \partial r}$
YX_{udot}	Υδροδυναμική παράγωγος	$\frac{\partial X}{\partial \dot{u}}$
YN_{vdot}	Υδροδυναμική παράγωγος	$\frac{\partial N}{\partial \dot{v}}$
YN_{rdot}	Υδροδυναμική παράγωγος	$\frac{\partial N}{\partial \dot{r}}$
YY_{v}	Υδροδυναμική παράγωγος	$\frac{\partial Y}{\partial v}$
YY_{r}	Υδροδυναμική παράγωγος	$\frac{\partial Y}{\partial r}$
YY_{vv}	Υδροδυναμική παράγωγος	$\frac{\partial^2 Y}{\partial v^2}$

YY_{vr}	Υδροδυναμική παράγωγος	$\frac{\partial^2 Y}{\partial v \partial r}$
YY_{rr}	Υδροδυναμική παράγωγος	$\frac{\partial^2 Y}{\partial r^2}$
YN_{r}	Υδροδυναμική παράγωγος	$\frac{\partial N}{\partial r}$
YN_{v}	Υδροδυναμική παράγωγος	$\frac{\partial N}{\partial v}$
YN_{rr}	Υδροδυναμική παράγωγος	$\frac{\partial^2 N}{\partial r^2}$
YN_{rrv}	Υδροδυναμική παράγωγος	$\frac{\partial^3 N}{\partial r^2 \partial v}$
YN_{vvr}	Υδροδυναμική παράγωγος	$\frac{\partial^3 N}{\partial v^2 \partial r}$
YN_{phi}	Υδροδυναμική παράγωγος	$\frac{\partial N}{\partial phi}$
YN_{vphi}	Υδροδυναμική παράγωγος	$\frac{\partial^2 N}{\partial v \partial phi}$
YN_{rphi}	Υδροδυναμική παράγωγος	$\frac{\partial^2 N}{\partial r \partial phi}$
YK_{phi}	Υδροδυναμική παράγωγος	$\frac{\partial K}{\partial phi}$
YK_{p}	Υδροδυναμική παράγωγος	$\frac{\partial K}{\partial p}$
zY	Καθ 'υψος κέντρο δράσης της υδροδυναμικής δύναμης Y	
YK_{pdot}	Υδροδυναμική παράγωγος	$\frac{\partial K}{\partial \dot{p}}$
wPO	Συντελεστής ομμορού έλικας	
wRO	Συντελεστής ομμορού πηδαλίου	
tPO	Συντελεστής μείωσης ώσης έλικας	
tRO	Συντελεστής μείωσης ώσης πηδαλίου	
sm	Μάζα του πλοίου	
xG	Διαμήκης απόσταση κ.β. του πλοίου από την μέση τομή	
zG	Καθ ύψος απόσταση κ.β. του πλοίου από την μέση τομή	
sIx	Ροπή αδράνειας ως πρός τον άξονα x	

sIz	Ροπή αδράνειας ως πρός τον άξονα z
aan	Στροφές έλικας
D	Διάμετρος έλικας
Pitch	Βήμα έλικας
xP	Απόσταση έλικας από μέση τομή
BLAR	Εκτεταμένη επιφάνεια έλικας
RAREA	Επιφάνεια πηδαλίου
alamda	Λόγος επιμήκους
HR	Ύψος πηδαλίου
xRO	Διαμήκης θέση κέντρου πίεσης πηδαλίου
zRO	Καθ 'υψος θέση κέντρου πίεσης πηδαλίου
sL	Μήκος Πλοίου
DRT	Βύθισμα πλοίου
DISP	Εκτόπισμα πλοίου
CM	Συντελεστής Μέσης Τομής
a1	Συντελεστής ανίστασης
a2	Συντελεστής ανίστασης
a3	Συντελεστής ανίστασης
cc1	Συντελεστής ώσης
c2	Συντελεστής ώσης
c3	Συντελεστής ώσης
WL	Μήκος κύματος
HW	Ύψος κύματος

ΠΙΝΑΚΑΣ 2 . Σταθερές εισαγωγής στο Header file shipspecs.h

Επίσης το αρχείο shipspecs.h περιέχει και τους όρους BA1, BA2, BA3, BB1, BB2, BB3, BC1, BC2, BC3 και DET. Πρόκειται για σταθερούς όρους (αποτελέσματα πράξεων μεταξύ των προηγουμένων όρων) που εισέρχονται στην έκφραση του συστήματος των διαφορικών εξισώσεων και χρησιμοποιούνται για λόγους ευκολίας κατά τον προγραμματισμό.
Το αρχείο shipspecs.h αποτελεί header file του συνολικού προγράμματος. Ο ορισμός αυτός γίνεται για προγραμματιστικούς και πρακτικούς λόγους. Στην συνέχεια του κώδικα με την δήλωση στην αρχή:

#include <shipspecs.h>

έχουμε ορισμένες και μπορούμε να χειριστούμε τις σταθερές που βρίσκονται μέσα στο αρχείο shipspecs.h.

Το Header file ουσιαστικά αποτελεί τον γενικό ορισμό του πλοίου (γεωμετρικό, υδροδυναμικό) που χρησιμοποιείται στο πρόγραμμα. Συνεπώς αν θέλαμε να χρησιμοποιήσουμε κάποιο άλλο πλοίο για προσομοίωση θα έπρεπε να εισάγουμε τα χαρακτηριστικά του στο πρόγραμμα επεμβαίνοντας στο header file. Για τον λόγο αυτό το header file είναι έτσι γραμμένο ώστε να είναι ευανάγνωστο και να μπορεί κάποιος χρήστης χωρίς εμπειρία στον προγραμματισμό να ορίσει τα χαρακτηριστικά του πλοίου που επιθυμεί να εισάγει στον προσομοιωτή. Στο shipspecs.h η κάθε κατηγορία γεωμετρικών και υδροδυναμικών χαρακτηριστικών είναι «χωρισμένη» από τις υπόλοιπες και υπάρχει τίτλος που δηλώνει σε ποια κατηγορία βρισκόμαστε. Πιο αναλυτικά οι βασικές κατηγορίες είναι:

- Κατηγορία mass properties: Τα βασικά χαρακτηριστικά της μάζας του πλοίου (κέντρα βάρους, ροπές αδράνειας κ.α.)

- Κατηγορία propeller properties: Τα βασικά χαρακτηριστικά της έλικας του πλοίου (διάμετρος, βήμα κ.α.)

- Κατηγορία rudder properties: Τα βασικά χαρακτηριστικά του πηδαλίου του πλοίου (επιφάνεια, κέντρα πίεσης κ.α.)

- Κατηγορία hydrodynamic coefficients 1: Υδροδυναμικές παράγωγοι.
- Κατηγορία hydrodynamic coefficients 2: Υδροδυναμικές παράγωγοι.
- Κατηγορία hydrodynamic coefficients3A: Υδροδυναμικές παράγωγοι.
- Κατηγορία hydrodynamic coefficients3B: Υδροδυναμικές παράγωγοι.
- Κατηγορία hydrodynamic coefficients4: Υδροδυναμικές παράγωγοι.
- Κατηγορία wake: Συντελεστές ομόρου
- Κατηγορία thrude: Συντελεστές μείωσης ώσης
- Κατηγορία ship data: βασικά χαρακτηριστικά του πλοίου
- Κατηγορία threshold: Συντελεστές αντίστασης,ώσης
- Κατηγορία wave: Χαρακτηριστικά κύματος (ανενέργη στην περίπτωσή μας).

B) Functions.cpp

Στο αρχείο functions.cpp βρίσκονται οι όλες οι συναρτήσεις που υπολογίζουν τις κινήσεις του πλοίου. Για τον υπολογισμό μιας κίνησης του πλοίου, π.χ. κίνηση κατά την διεύθυνση περιστροφής roll, χρησιμοποιούνται τρεις ξεχωριστές συναρτήσεις. Μια υπολογίζει την επίδραση της έλικας, μια της γάστρας και μία του πηδαλίου. Όλες οι

συναρτήσεις χρησιμοποιούν στους υπολογισμούς τα δεδομένα (γεωμετρικά, υδροδυναμικά χαρακτηριστικά του πλοίου) που βρίσκονται στο Header file shipspecs.h. Τέλος υπάρχει η τελική συνάρτηση που χρησιμοποιεί τα αποτελέσματα των άλλων τριών συναρτήσεων και υπολογίζει τις ζητούμενες συνολικές δυνάμεις και ροπές που ασκούνται στο πλοίο.

Οι βασικές παράμετροι που εισέρχονται στις συναρτήσεις του functions.cpp είναι:

- Παράμετρος u: Η κίνηση του πλοίου κατά την διεύθυνση του άξονα των x του σωματοπαγούς συστήματος συντεταγμένων του πλοίου (κίνηση surge).

- Παράμετρος v: Η κίνηση του πλοίου κατά την διεύθυνση του άξονα των y του σωματοπαγούς συστήματος συντεταγμένων του πλοίου (κίνηση sway).

- Παράμετρος r: Η ρυθμός περιστροφής του πλοίου γύρω από τον άξονα των z του σωματοπαγούς συστήματος συντεταγμένων του πλοίου (κίνηση yaw).

- Παράμετρος p: Ο ρυθμός περιστροφής του πλοίου γύρω από τον άξονα των x του σωματοπαγούς συστήματος συντεταγμένων του πλοίου (κίνηση roll).

- Παράμετρος phi: Η γωνία εγκάρσιας κλίσης του πλοίου κατά την διάρκεια στροφής (απουσία κυματισμών).

- Παράμετρος psi: Η γωνία που σχηματίζει η προέκταση του άξονα των x του σωματοπαγούς συστήματος συντεταγμένων του πλοίου με τον άξονα των y του επίγειου συστήματος συντεταγμένων.

- Παραμετρος xi: Η x-συντεταγμένη της θέσης του πλοίου ως προς το επίγειο σύστημα συντεταγμένων.

- Παράμετρος yi: Η y-συντεταγμένη της θέσης του πλοίου ως προς το επίγειο σύστημα συντεταγμένων.

- Παράμετρος delta: Η γωνία εκτροπής του πηδαλίου σε rad.

Οι συναρτήσεις που βρίσκονται στο αρχείο functions.cpp και υπολογίζουν τις δυνάμεις και ροπές που ασκούνται στο πλοίο είναι:

- Συνάρτηση Fxp: υπολογίζει την δύναμη σε κίνηση surge τής έλικας και δέχεται ως παράμετρο τις μεταβλητές u,v,r.

- Συνάρτηση Fxh: υπολογίζει την δύναμη σε κίνηση surge τής γάστρας και δέχεται ως παραμέτρους τις μεταβλητές u,v,r.

- Συνάρτηση Fxr: υπολογίζει την δύναμη σε κίνηση surge του πηδαλίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta.

- Συνάρτηση F1: υπολογίζει την συνολική δύναμη σε κίνηση surge που επιδρά στο πλοίο και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta,p,phi,psi,xi. Η συνάρτηση F1 χρησιμοποιεί τα αποτελέσματα των Fxp, Fxr, Fxh όπως φαίνεται και στον κώδικα που παρατίθεται στο παράρτημα.

- Συνάρτηση Fyp: υπολογίζει την δύναμη σε κίνηση sway της έλικας και δέχεται ως παράμετρο την μεταβλητή u.

- Συνάρτηση Fyh: υπολογίζει την δύναμη σε κίνηση sway της γάστρας και δέχεται ως παραμέτρους τις μεταβλητές u,v,r.

- Συνάρτηση Fyr: υπολογίζει την δύναμη σε κίνηση sway του πηδαλίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta.

- Συνάρτηση F2: υπολογίζει την συνολική δύναμη σε κίνηση sway που επιδρά στο πλοίο και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta,p,phi,psi,xi. Η συνάρτηση F2 χρησιμοποιεί τα αποτελέσματα των Fyp, Fyr, Fyh όπως φαίνεται και στον κώδικα που παρατίθεται στο παράρτημα.

- Συνάρτηση Fnh: υπολογίζει την ροπή σε κίνηση yaw της γάστρας του πλοίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,phi.

- Συνάρτηση Fnp: υπολογίζει την ροπή σε κίνηση yaw τής έλικας και δέχεται ως παράμετρο την μεταβλητή u.

- Συνάρτηση Fnr: υπολογίζει την ροπή σε κίνηση yaw του πηδαλίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta.

- Συνάρτηση F3: υπολογίζει την συνολική ροπή σε κίνηση yaw που επιδρά στο πλοίο και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta,p,phi,psi,xi. Η συνάρτηση F3 χρησιμοποιεί τα αποτελέσματα των Fnp, Fnr, Fnh όπως φαίνεται και στον κώδικα που παρατίθεται στο παράρτημα.

- Συνάρτηση Fkh: υπολογίζει την ροπή σε κίνηση roll της γάστρας του πλοίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,p,phi,xi.

- Συνάρτηση Fkp: υπολογίζει την ροπή σε κίνηση roll τής έλικας και δέχεται ως παράμετρο την μεταβλητή u.

- Συνάρτηση Fkr: υπολογίζει την ροπή σε κίνηση roll του πηδαλίου και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta.

- Συνάρτηση F4: υπολογίζει την συνολική ροπή σε κίνηση roll που επιδρά στο πλοίο και δέχεται ως παραμέτρους τις μεταβλητές u,v,r,delta,p,phi,psi,xi. Η συνάρτηση F4 χρησιμοποιεί τα αποτελέσματα των Fkp, Fkr, Fkh όπως φαίνεται και στον κώδικα που παρατίθεται στο παράρτημα.

Οι συναρτήσεις F1, F2, F3, F4 οι οποίες βρίσκονται στο αρχείο functions.cpp και υπολογίζουν τις συνολικές δυνάμεις και ροπές που ενεργούν επί του πλοίου παίρνουν ως παραμέτρους τις u,v,r,delta,p,phi,psi,xi και όχι την παράμετρο yi. Επίσης,όπως φαίνεται και στον κώδικα που παρατίθεται στο παράρτημα αρκετές συναρτήσεις δεν χρησιμοποιούν στους υπολογισμούς όλες τις παραμέτρους που εισέρχονται σ'αυτές. Αυτό συμβαίνει κυρίως για να είναι εύχρηστη η ρουτίνα imsl_d_ode_runge_kutta της βιβλιοθήκης IMCL.C που απαιτεί μια ομοιομορφία, από άποψη παραμέτρων, για τις συναρτήσεις που αποτελούν το σύστημα των διαφορικών εξισώσεων.

Όλες οι παραπάνω συναρτήσεις βασίζονται στο θεωρητικό μοντέλο που περιγράψαμε στο κεφάλαιο ΙΙ. Αν μελετήσουμε τον κώδικα που βρίσκεται στο παράρτημα θα παρατηρήσουμε μερικές διαφορές μεταξύ των συναρτήσεων του αρχείου functions.cpp και των αντίστοιχων σχέσεων που παρουσιάζονται στο κεφάλαιο ΙΙ. Οι διαφορές αυτές αφορούν την παράληψη των παραγώγων των μεγεθών u,v,r,p που εισέρχονται στις σχέσεις του κεφαλαίου ΙΙ. Παρ' όλα αυτά, η ακρίβεια των αποτελεσμάτων είναι αρκετά ικανοποιητική.

Στο αρχείο functions.cpp υπάρχει ορισμένη και η συνάρτηση fcn. Η συνάρτηση αυτή, λειτουργεί για λογαριασμό της ρουτίνας imsl_d_ode_runge_kutta της βιβλιοθήκης IMSL.C. Εισάγει τις αρχικές

τιμές των μεγεθών u,v,r,delta,p,phi,psi,xi,yi (που ορίζουμε εμείς) στις συναρτήσεις F1, F2, F3 και F4 και ορίζει το σύστημα των διαφορικών εξισώσεων που επιθυμούμε να λύσουμε με την αριθμητική επαναληπτική μέθοδο Runge-Kutta 4ου βαθμού.

C) Simulation.cpp

Σ' αυτό το αρχείο βρίσκεται το τμήμα το προγράμματος που αναλαμβάνει την απεικόνιση των τρισδιάστατων γραφικών, την κίνησή τους και την δυνατότητα αλληλεπίδρασης του χρήστη σε πραγματικό χρόνο. Είναι προφανές ότι ο κώδικας που αποτελεί το αρχείο simulation.cpp είναι το εκτελεστικό μέρος του προγράμματος (εδώ περιέχεται η main). Σε αυτή τη φάση χρησιμοποιούνται οι βιβλιοθήκες Coin3d (Open Inventor) και IMSL.C. Οπώς αναφέρθηκε και προηγουμένως η βιβλιοθήκη IMSL.C χρησιμοποιείται για την αριθμητική επίλυση του συστήματος των διαφορικών εξισώσεων με την επαναληπτική μέθοδο Runge-Kutta 4ου βαθμού. Η βιβλιοθήκη Coin3d (Open Inventor) χρησιμοποιείται για την εισαγωγή των αντικειμένων (πλοίο, θάλασσα, λιμάνι) στο πρόγραμμα, τον ορισμό των κινήσεών τους σε πραγματικό χρόνο και την επέμβαση του χρήστη στο εικονικό περιβάλλον.
Αξίζει να σημειωθεί ότι όταν αναφερόμαστε σε πραγματικό χρόνο (real time) αναφερόμαστε στον χρόνο που μετρά ο υπολογιστής. Ο χρόνος αυτός διαφέρει από τον δικό μας χρόνο και προφανώς υστερεί κάπως. Σε περιπτώσεις υπερφόρτωσης του επεξεργαστή η υστέρηση αυτή μπορεί να είναι πιο εμφανής γι' αυτό ένας καλός επεξεργαστής είναι απαραίτητος για απεικονίσεις και υπολογισμούς σε πραγματικό χρόνο. Σε γενικές γραμμές όμως η αναπαράσταση του χρόνου κρίνεται αρκετά ικανοποιητική.

Ο κώδικας εξάγει όλα τα αποτελέσματα των επιλύσεων του συστήματος των διαφορικών εξισώσεων σε ένα αρχείο notepad που ονομάζεται

simoutput.txt. Τα αποτελέσματα μπορούν εύκολα στην συνέχεια να εισαχθούν σε άλλα προγράμματα (autocad, excel, rhino) και να γίνει αξιολόγηση και περαιτέρω επεξεργασία τους.

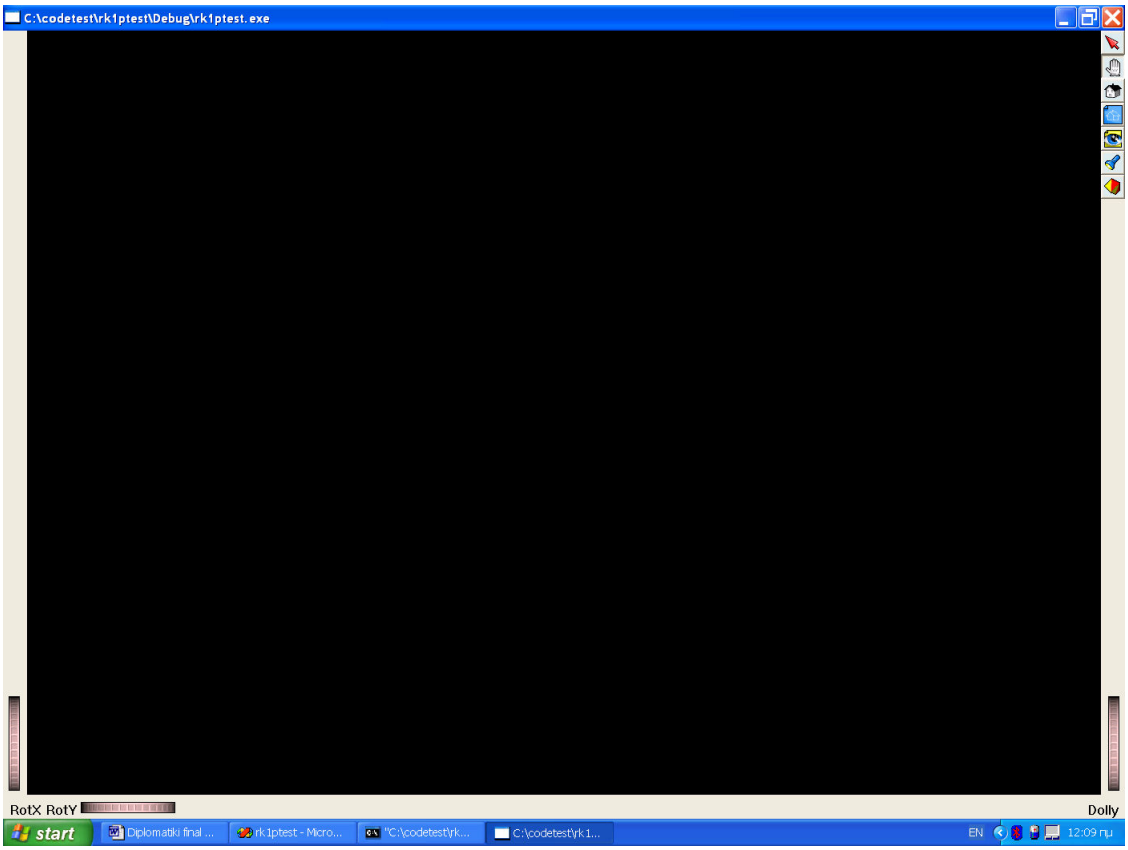
Στην συνέχεια θα περιγράψουμε τον κώδικα χωρίς να εμβαθύνουμε στην επεξήγηση της ορολογίας και της σύνταξης της γλώσσας προγραμματισμού C++.Ο κώδικας παρατίθεται στο παράρτημα.

Αρχικά, δημιουργείται ένα παράθυρο windows μέσα στο οποίο θα ορίσουμε τα αντικείμενα που επιθυμούμε να δημιουργήσουμε (να «ζωγραφίσει» η μηχανή γραφικών του υπολογιστή). Επίσης θα ορίσουμε και τις κινήσεις στις οποίες θα υπακούουν τα αντικείμενα. Στο παράθυρο αυτό πρέπει να οριστούν ακόμη η κάμερα μέσα από την οποία θα γίνει η παρατήρηση καθώς και διάφορες άλλοι παράμετροι πού αφορούν τον φωτισμό,την εμφάνιση του υλικού των αντικείμενων κ.α. Στην ορολογία της βιβλιοθήκης Open Inventor όλα τα παραπάνω που αφορούν την αναπαράσταση μέσα στο παράθυρο των windows ονομάζονται Scene graph και αυτό τον όρο θα χρησιμοποιούμε στο μέλλον.
Για να δημιουργήσουμε το παράθυρο δημιουργούμε ένα class SoWinExaminerViewer το οποίο ονομάζουμε viewer με την εντολή: SoWinExaminerViewer * viewer = new SoWinExaminerViewer(window);

Το class αυτό διαθέτει την συνάρτηση που δημιουργεί το παράθυρο μέσα στο οποίο στην συνέχεια θα ορίσουμε το Scene graph. Στο τέλος του κώδικα, για να εμφανιστεί το παράθυρο και ότι έχουμε ορίσει μέσα σε αυτό καλούμε την συνάρτηση show του class SoWin με παράμετρο το παράθυρο που δημιουργήσαμε στην αρχή.

```
SoWin::show(window);
```

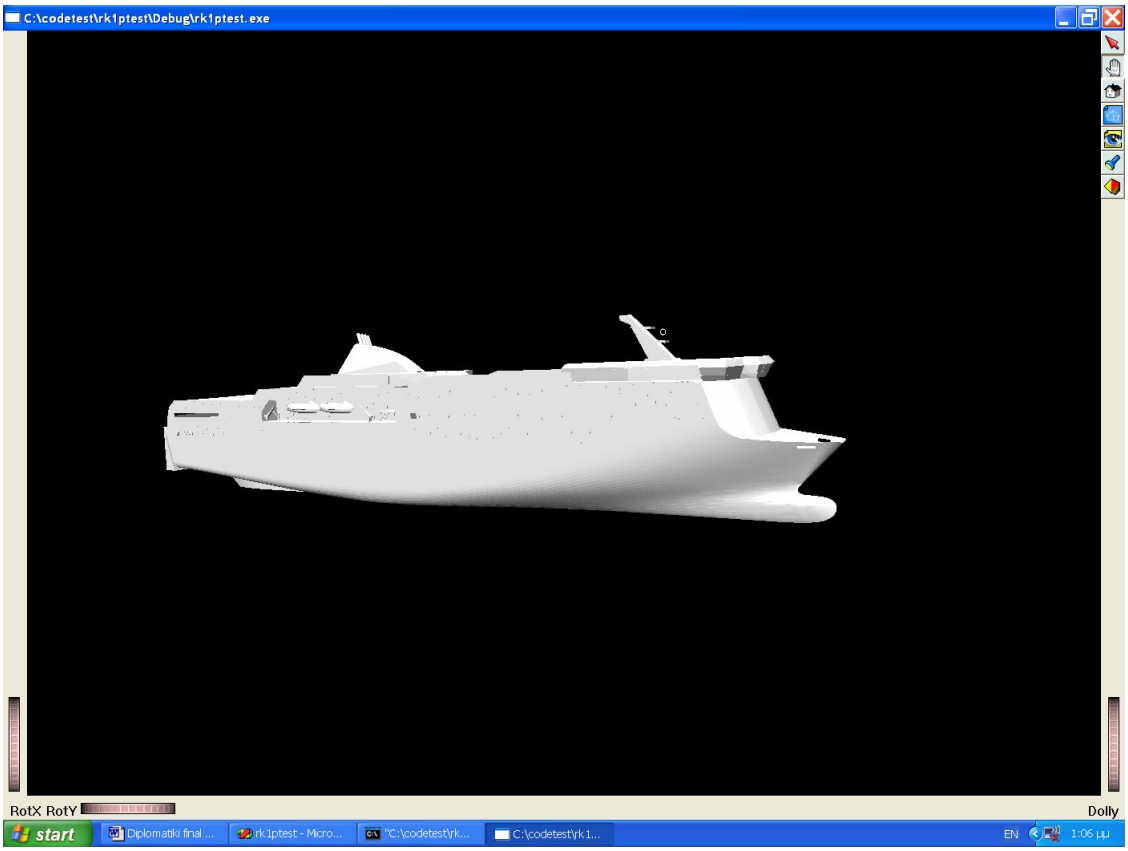
Το παράθυρο το οποίο τελικώς θα δημιουργηθεί διαθέτει και την δυνατότητα να γίνονται ορισμένοι χειρισμοί στην κάμερα παρατήρησης κατά την διάρκεια της προσομοίωσης. Οι σημαντικότεροι είναι η περιστροφή της κάμερας κατά τους άξονες x και y (του επίγειου συστήματος συντεταγμένων) και η δυνατότητα για zoom in και zoom out. Το κενό παράθυρο (χωρίς το scene graph) φαίνεται στην παρακάτω εικόνα (Εικόνα 1).



Εικόνα 1. Examiner viewer με κενό scene graph

Στον κώδικα εισέρχεται (import) το τρισδιάστατο μοντέλο του πλοίου που βρίσκεται σε αρχείο με την κατάληξη .wrml , .wrl . Πρόκειται για το επιβατηγό – οχηματαγωγό Αλκυών του οποίου γνωρίζουμε όλα τα γεωμετρικά και υδροδυναμικά χαρακτηριστικά τα οποία έχουμε εισάγει στο header file shipspecs.h. Προφανώς πάνω σε αυτό το “αντικείμενο” θα εφαρμοστούν τα αποτελέσματα του μαθηματικού μοντέλου με την επίλυση

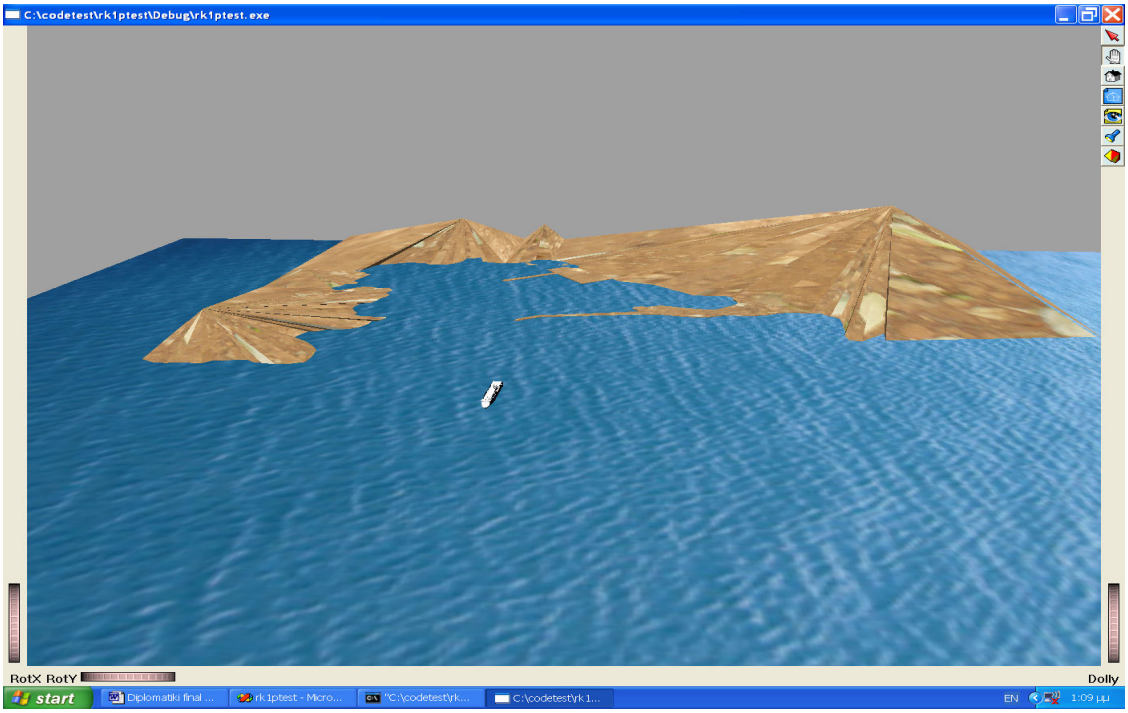
του συστήματος των διαφορικών εξισώσεων που διέπουν το φαινόμενο. (Εικόνα 2).



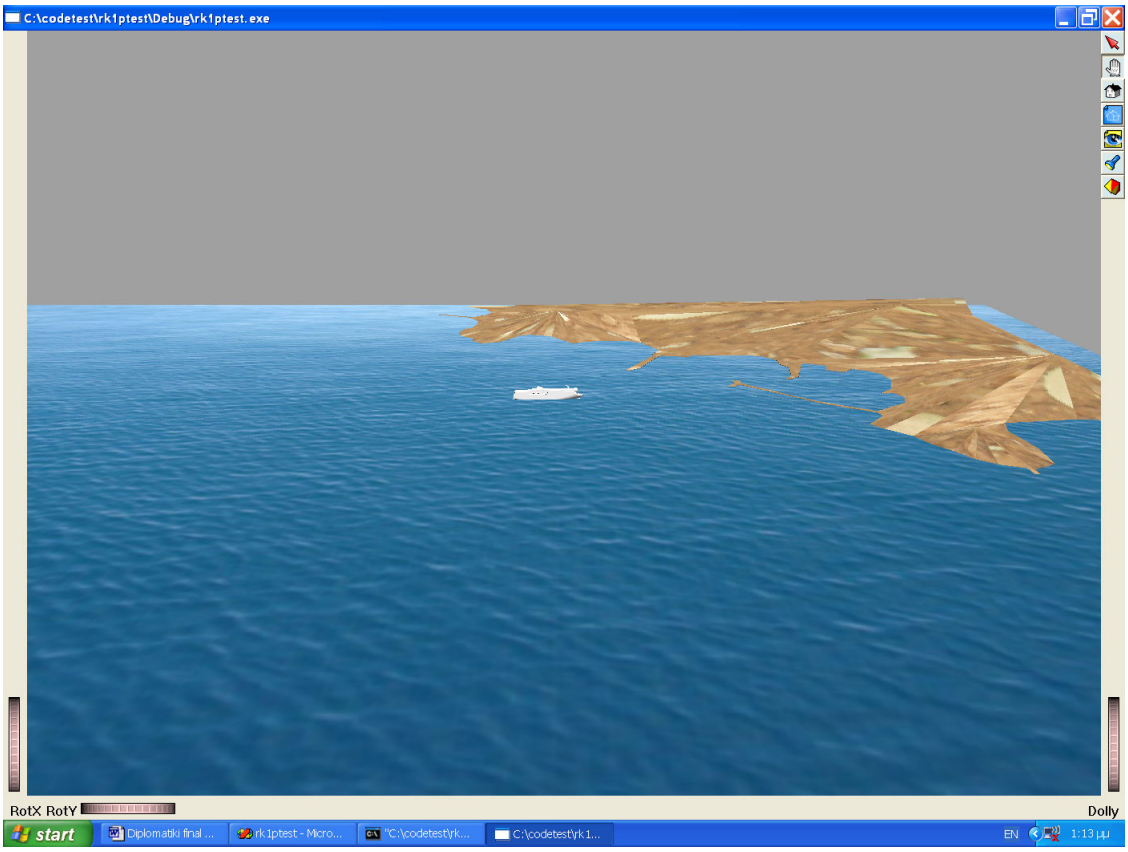
Εικόνα 2. Το πλοίο Αλκυών στο scene graph

Επίσης, εισέρχεται και η επιφάνεια της θάλασσας και το τρισδιάστατο μοντέλο του λιμανιού της Σύρου. Υπάρχει η δυνατότητα να εισαχθεί στο πρόγραμμα και η τρισδιάστατη απεικόνιση του λιμανιού της Τήνου με μια

απλή μετονομασία του αρχείου που αναφέρουμε στον κώδικα. Αντί για syros.wrl τυπώνουμε tinos.wrl (Εικόνα 3, Εικόνα 4).



Εικόνα 3. Τελικό scene graph 1:Πλοίο Αλκυών, θάλασσα και λιμάνι Σύρου.



Εικόνα 4. Τελικό scene graph 2:Πλοίο Αλκυών, θάλασσα και λιμάνι Τήνου.

Όλα τα παραπάνω γίνονται δημιουργώντας ένα class SoInput και ορίζοντας το εξωτερικό αρχείο που θέλουμε να εισάγουμε στο Scene graph.

Να σημειωθεί ότι αρχεία με την κατάληξη .wrml , .wrl μπορούν εύκολα να δημιουργηθούν με πολλά CAD-CAM προγράμματα που υποστηρίζουν αυτό το file format (π.χ. Rinoceros). Με αυτόν τον τρόπο είναι δυνατή η χρησιμοποίηση (και εκμετάλλευση) πολύπλοκων γεωμετριών μέσα στον Open Inventor που προηγουμένως έχουν σχεδιαστεί σε ένα σχεδιαστικό πρόγραμμα το οποίο είναι σε θέση να ικανοποιήσει αυξημένες σχεδιαστικές και γεωμετρικές απαιτήσεις. Στο επόμενο κεφάλαιο περιγράφουμε την διαδικασία δημιουργίας των θαλασσών και λιμανιών.

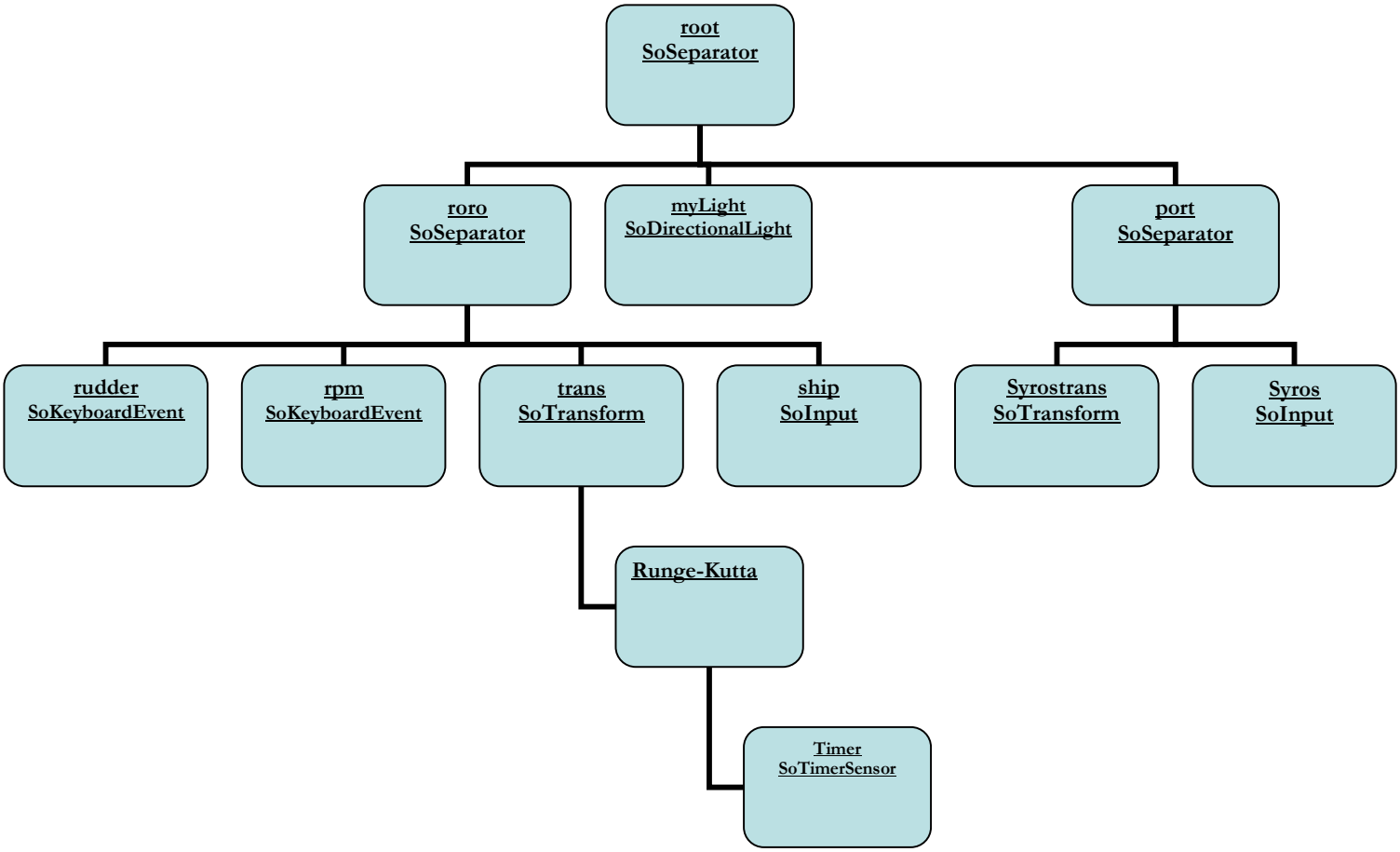
Το επιθυμητό scene graph που προσπαθούμε να δημιουργήσουμε αποτελείται από :

- Το πλοίο Αλκυών να εκτελεί τις κινήσεις (αποκρίσεις) του και να το κυβερνούμε μεταβάλλοντας με πλήκτρα την γωνία του πηδαλίου και τις στροφές της έλικας.

- Την επιφάνεια της θάλασσας και τη γεωμετρία του λιμανιού.

Για την επίτευξη του στόχου μας κατασκευάζουμε τρεις "ξεχωριστές" κατηγορίες αντικειμένων-κινήσεων δημιουργώντας τρία classes So Separator. Το πρώτο το ονομάζουμε roro και θα αποτελεί το πλοίο Αλκυών με τις κινήσεις του και τις εξωτερικές επιβολές του χρήστη. Το δεύτερο το ονομάζουμε port και περιέχει τις γεωμετρίες της θάλασσας και του λιμανιού. Το τρίτο ονομάζεται root και το δημιουργήσαμε για να εισάγουμε σε αυτό τις δύο παραπάνω κατηγορίες ώστε τελικώς να εμφανιστούν όλα μέσα στο scene graph.

Στο γράφημα της επόμενης σελίδας (Σχήμα 2) παρουσιάζεται η φιλοσοφία της δομής του κώδικα. Εμφανίζονται τα classes τα οποία δημιουργούνται και παρουσιάζεται ο τρόπος αλληλεπίδρασής τους ώστε να καταλήξουμε στο τελικό επιθυμητό αποτέλεσμα στο root, όπου ορίζεται τι θα αναπαρασταθεί.



Σχήμα 2. Η δομή του κώδικα προσομοίωσης

Η επίτευξη του στόχου της αναπαράστασης και της αλληλεπίδρασης του χρήστη σε πραγματικό χρόνο εκτελείται ως εξής:
Χρησιμοποιώντας μια συνάρτηση από το class (SoTimeSensor) καλούμε κάθε δt second (στην περίπτωσή μας 0.1) την ρουτίνα imsl_d_ode_runge_kutta που βρίσκεται στην βιβλιοθήκη IMSL.C για να λύσει το σύστημα των διαφορικών εξισώσεων την συγκεκριμένη χρονική στιγμή t. Με χρήση άλλων συναρτήσεων από διαφορετικά classes (SoTransform για μετατοπίσεις, SoRotate για περιστροφές) εφαρμόζουμε καταλλήλως τα αποτελέσματα που επιστρέφει η ρουτίνα imsl_d_ode_runge_kutta ώστε να μπορούμε να παρατηρήσουμε τις αποκρίσεις του πλοίου μέσα από το παράθυρο που δημιουργήσαμε κατά την διάρκεια του χρόνου.

Στην συνέχεια δημιουργούμε ένα class SoKeyboardEvent και ορίζουμε τα πλήκτρα τα οποία θα καθορίζουν την αλληλεπίδραση του χρήστη με το περιβάλλον. Έτσι, όταν ο χρήστης με το πάτημα του πλήκτρου μεταβάλλει την τιμή της γωνίας του πηδαλίου στο επόμενο 0.1 second η ρουτίνα imsl_d_ode_runge_kutta θα λύσει το σύστημα των διαφορικών εξισώσεων με την νέα τιμή της γωνίας πηδαλίου και οι μεταβολές της κινητικής κατάστασης του πλοίου θα εφαρμοστούν στο “αντικείμενο” πλοίο .

Στην συγκεκριμένη περίπτωση οι δυνατότητες αλληλεπίδρασης που έχει ο χρήστης είναι:

Στροφή του πηδαλίου δεξιά και αριστερά κατά 0.1 rad με τα κουμπιά A για αριστερά και D για δεξιά.

Αύξηση και μείωση της ταχύτητας του πλοίου μεταβάλλοντας τις στροφές της έλικας aan κατά 0.5 rps με τα κουμπιά W για αύξηση και S για μείωση.

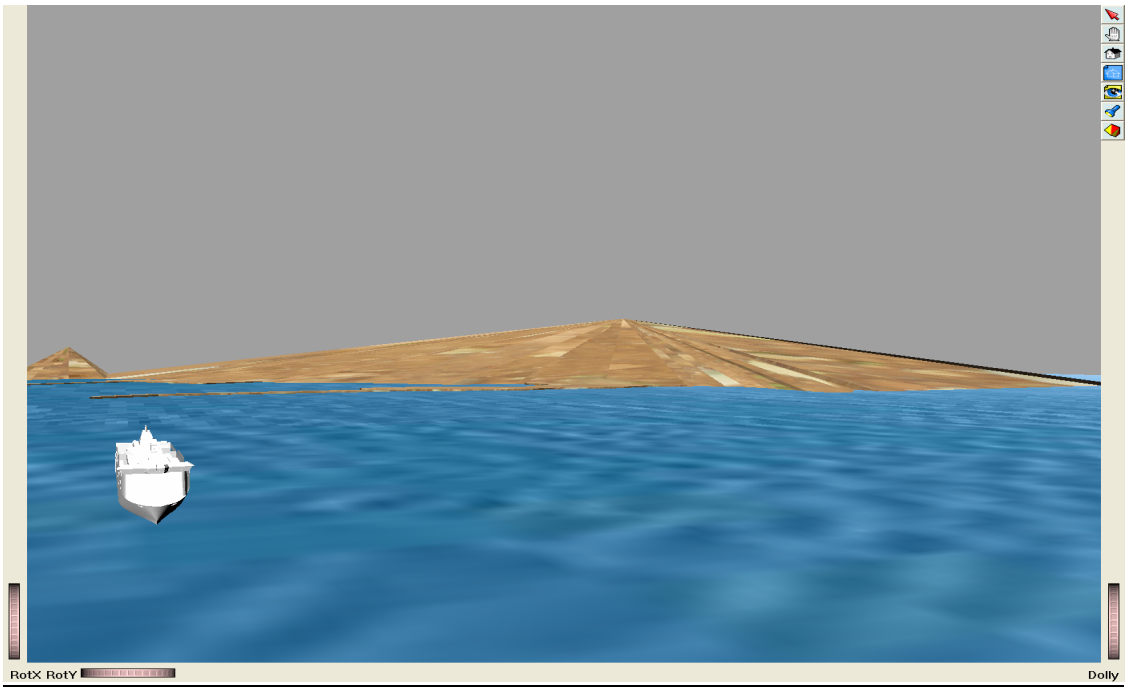
Με κάθε μεταβολή της γωνίας του πηδαλίου ή των στροφών της έλικας η νέα τιμή της μεταβλητής τυπώνεται στο παράθυρο MSDOS το οποίο βρίσκεται πίσω από το παράθυρο παρατήρησης που έχουμε δημιουργήσει.

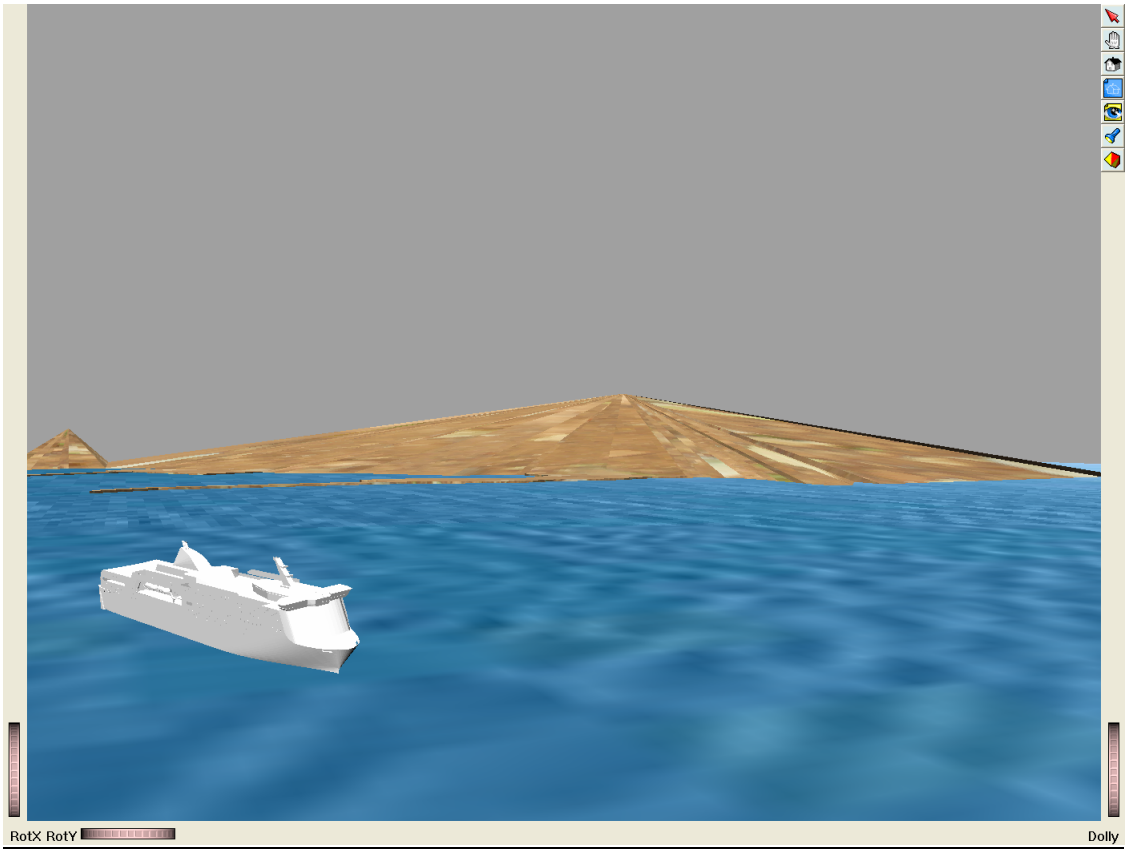
Στο scene graph που εκτελείται χρησιμοποιούμε και το Class SoDirectionalLight το οποίο, όπως ορίζει και η ονομασία του, δημιουργεί μία δέσμη φωτός με άπειρες διαστάσεις που χρησιμοποιείται συνήθως για την προσομοίωση του φωτός του ηλίου. Έτσι ορίζεται η κατεύθυνση της δέσμης που στην συγκεκριμένη περίπτώση είναι προς τα αρνητικά του άξονα Z ώστε να αντιπροσωπεύει τις 12.00 το μεσημέρι. Επίσης ορίζεται και το χρώμα της δέσμης που φυσικά είναι το λευκό. Αξίζει να σημειωθεί ότι με την χρησιμοποίηση άλλων class (π.χ. SoFog) υπάρχει η δυνατότητα να ορίσει ο χρήστης αν θα έχουμε μια ηλιόλουστη μέρα ή με συννεφιά, ομίχλη και τον βαθμό αυτής. Στην δική μας εφαρμογή, για λόγους απλότητας επιλέξαμε μία ηλιόλουστη μέρα χωρίς σύννεφα, ομίχλη και ότι άλλο επηρεάζει στην πραγματικότητα την ορατότητα.

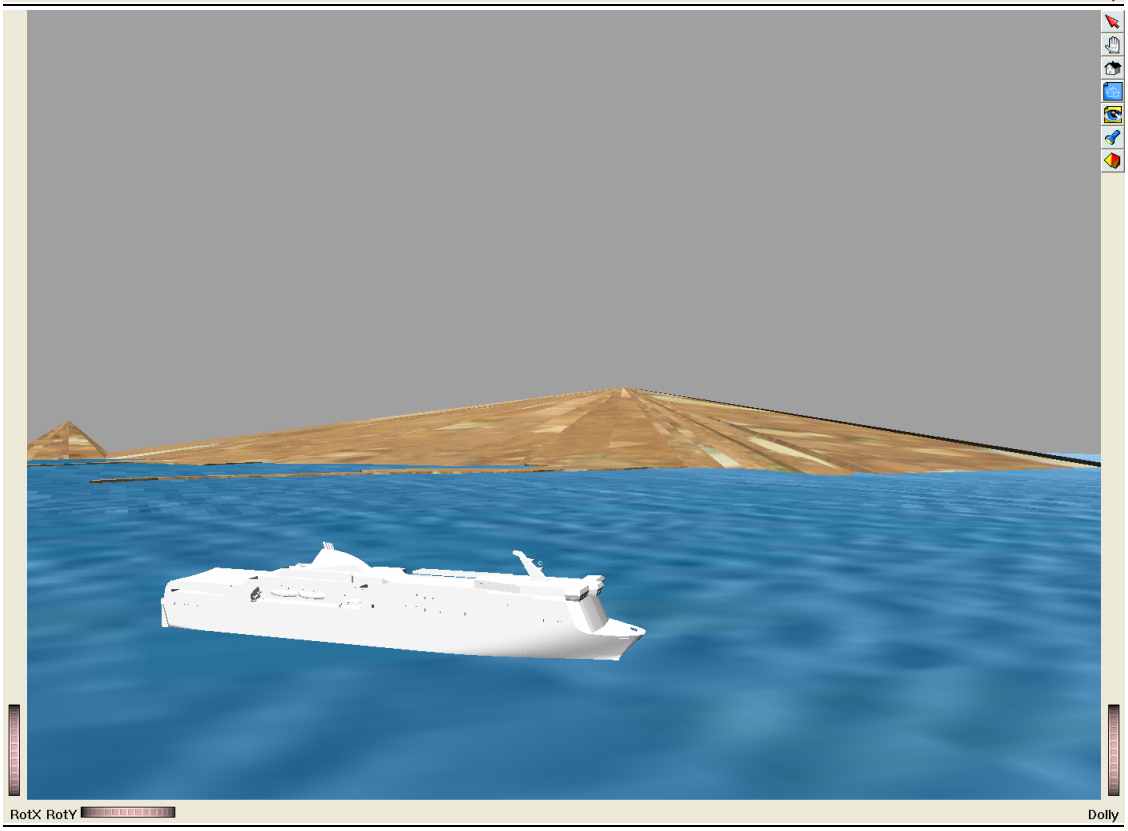
Όπως αναφέρθηκε προηγουμένως τα αποτελέσματα της επίλυσης του συστήματος των διαφορικών εξισώσεων κάθε 0,1 sec εξάγονται από τον κώδικα σε ένα αρχείο notepad με όνομα simoutput.txt. Στο αρχείο αυτό υπάρχουν όλα τα στοιχεία της “ιστορίας” της προσομοίωσης που μόλις εκτελέστηκε. Τα αποτελέσματα αυτά από το αρχείο simoutput.txt μπορούν εύκολα να εισαχθούν σε άλλα προγράμματα (excel,Rhinoceros,3DMAX κ.α.) για αξιολόγηση και περαιτέρω ανάλυση.

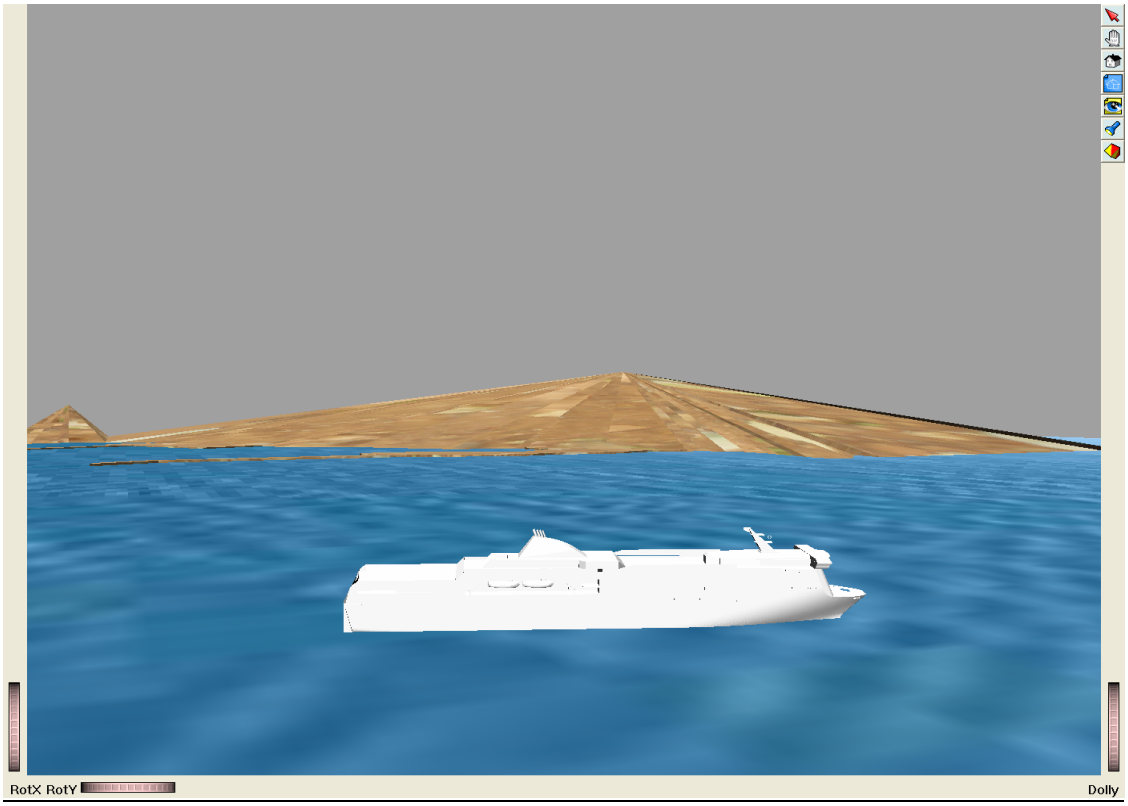
Στις παρακάτω σελίδες παρουσιάζεται η δοκιμή του αριστερού κύκλου στροφής του επιβατηγού/οχηματαγωγού Αλκυών με εκτροπή πηδαλίου κατά 0.5 rad και στροφές έλικας 3.39 rps που αντιστοιχούν σε ταχύτητα πλοίου ίση με 15.8 m/sec περίπου 30 κόμβοι. Τα αποτελέσματα της προσομοίωσης εισήχθησαν από το αρχείο simoutput.txt στο Excel και

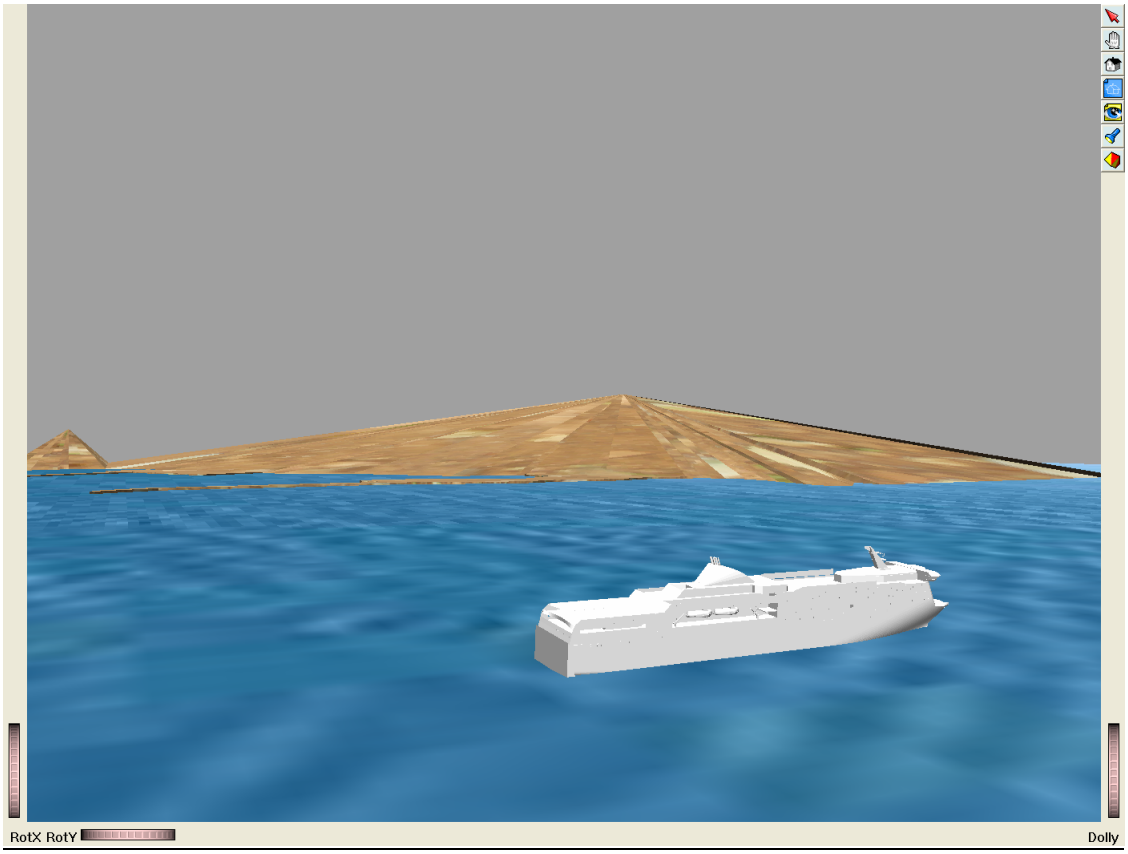
δημιουργήθηκαν οι γραφικές παραστάσεις της τροχιάς του πλοίου και της γωνίας εγκάρσιας κλίσης:

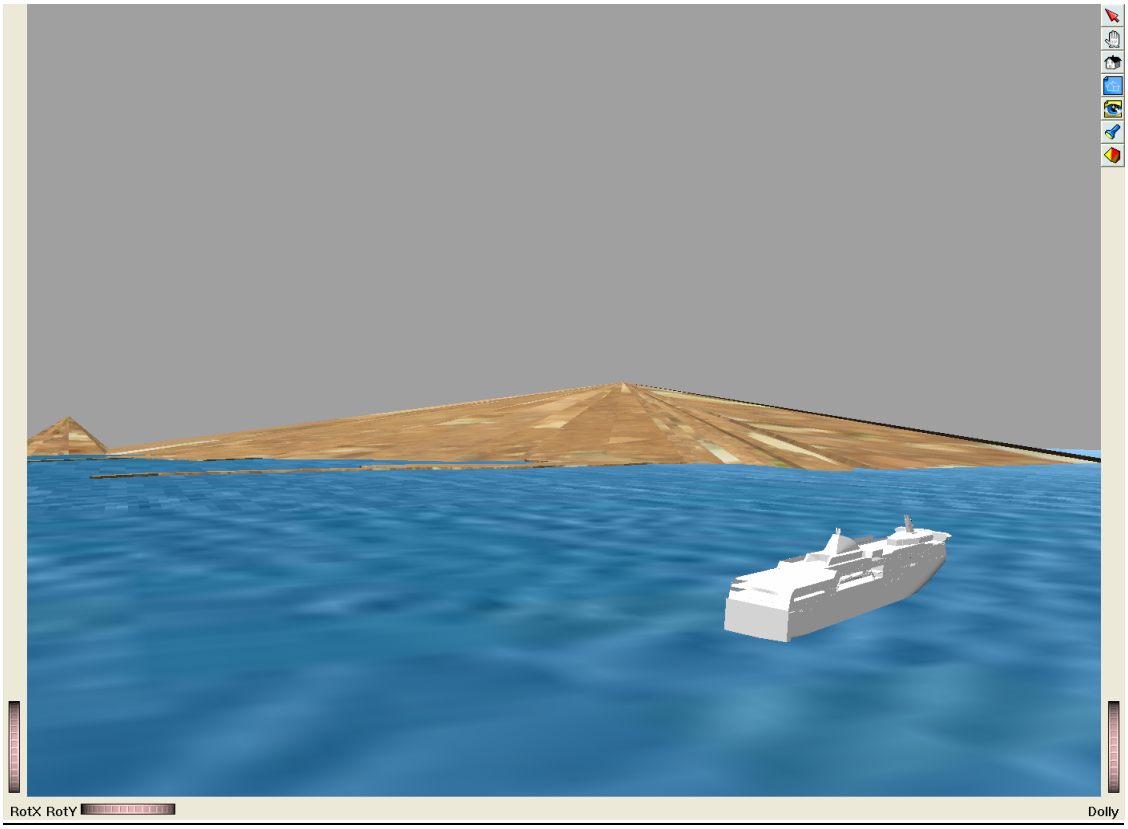


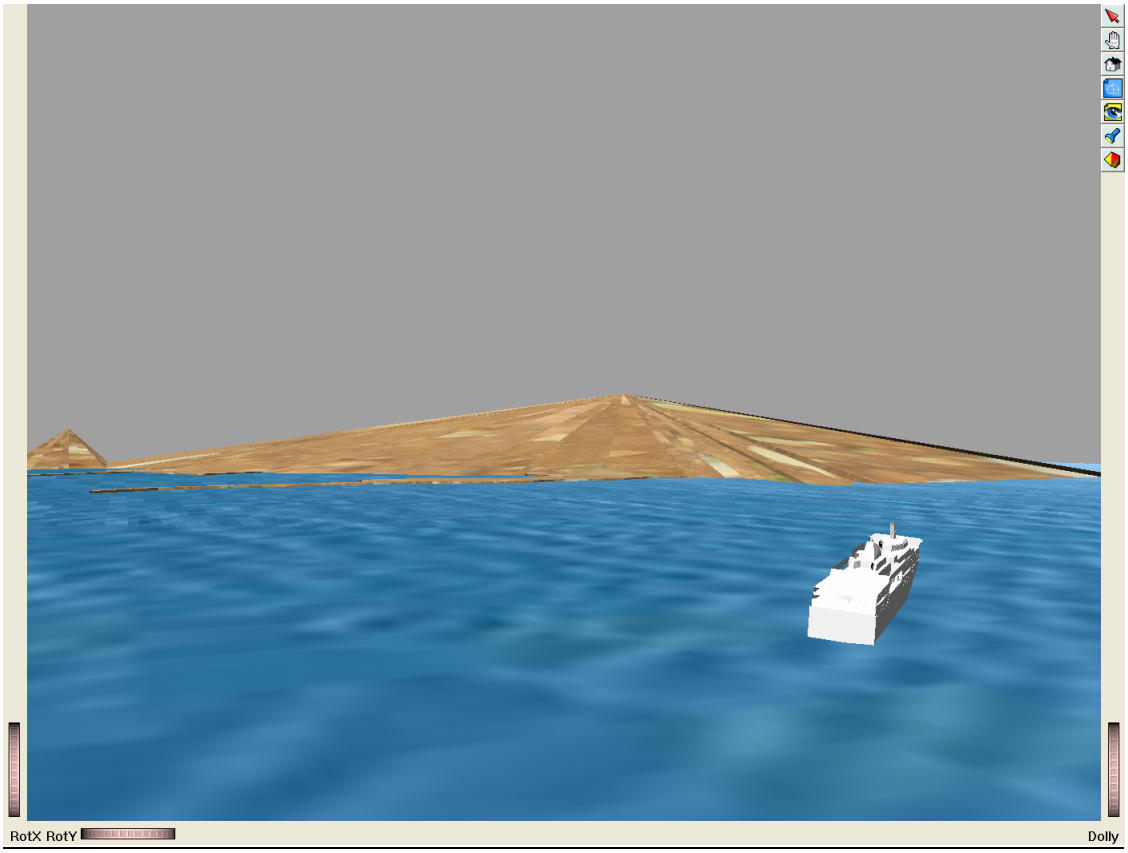

RotX RotY
Dolly

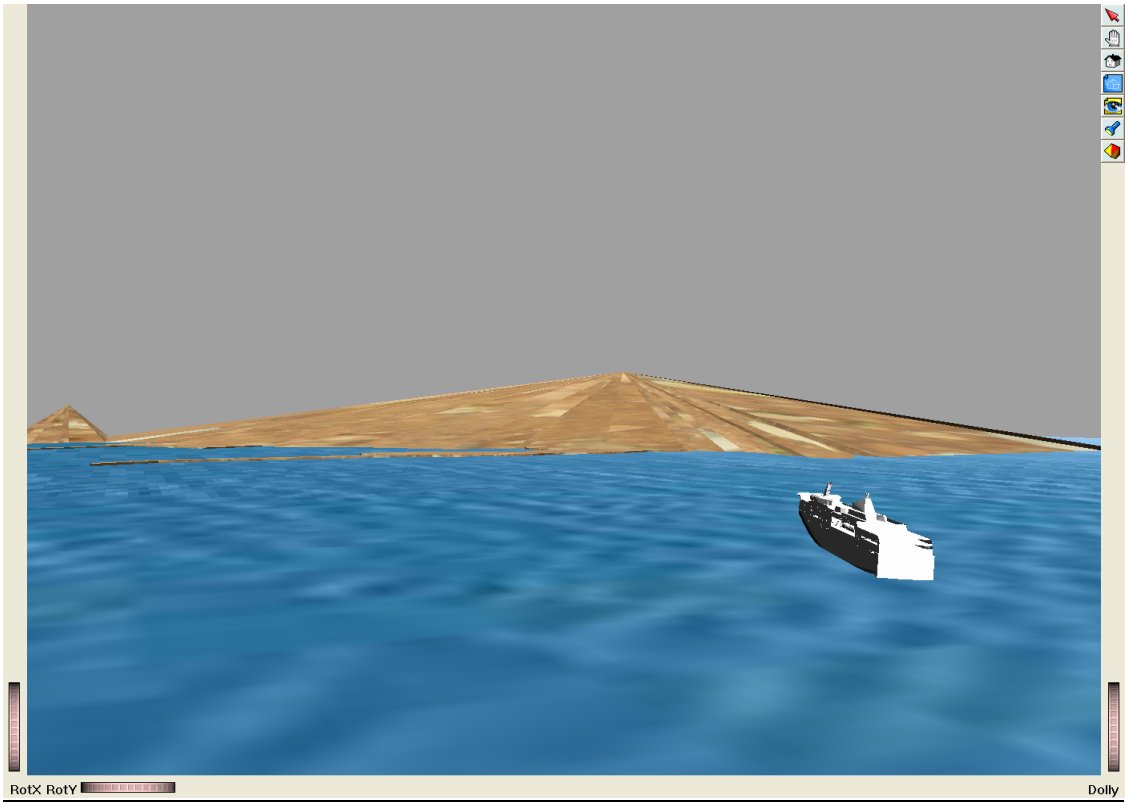

RotX RotY
Dolly

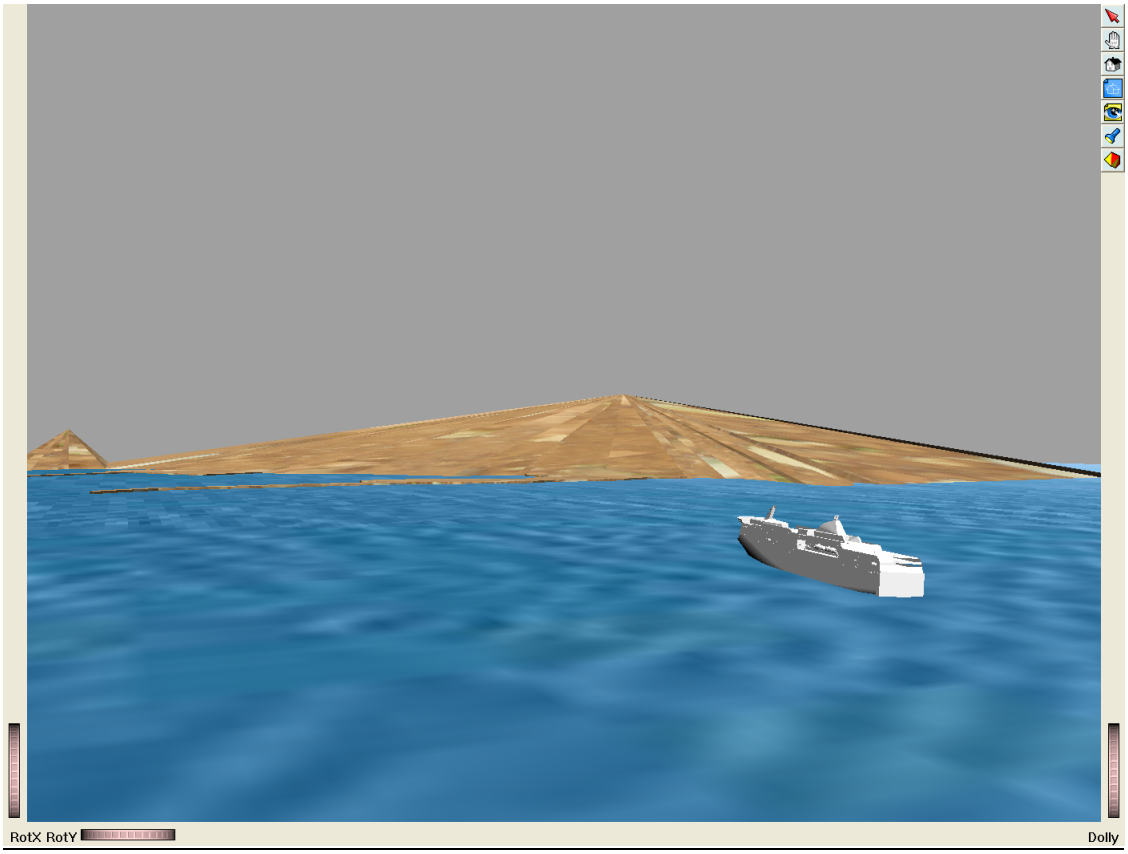

RotX RotY
Dolly

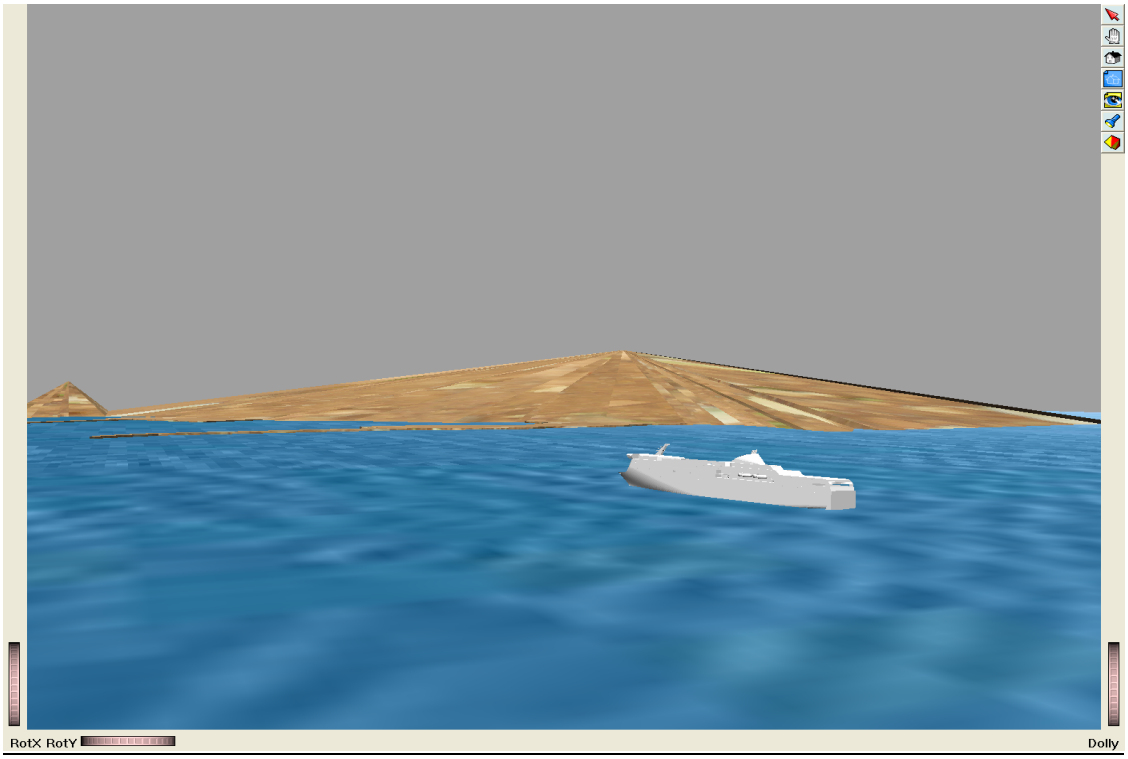

RotX RotY
Dolly

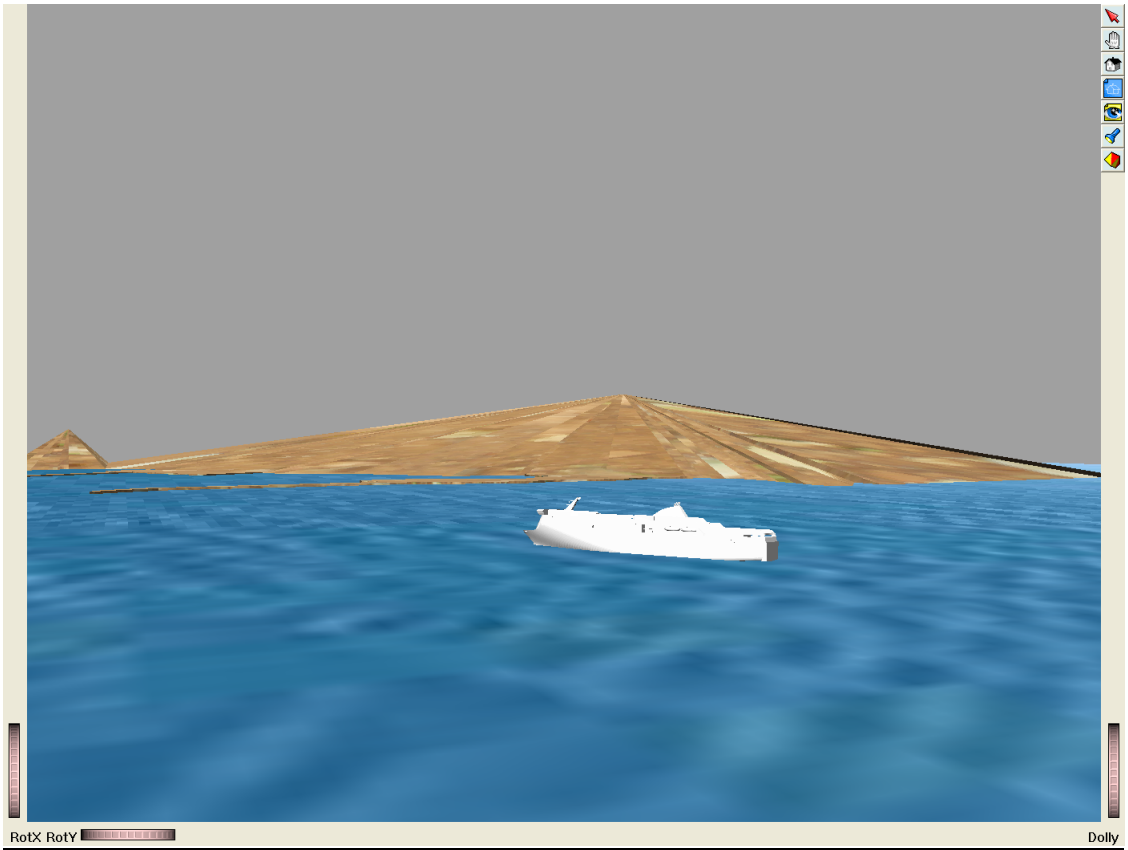

RotX RotY
Dolly

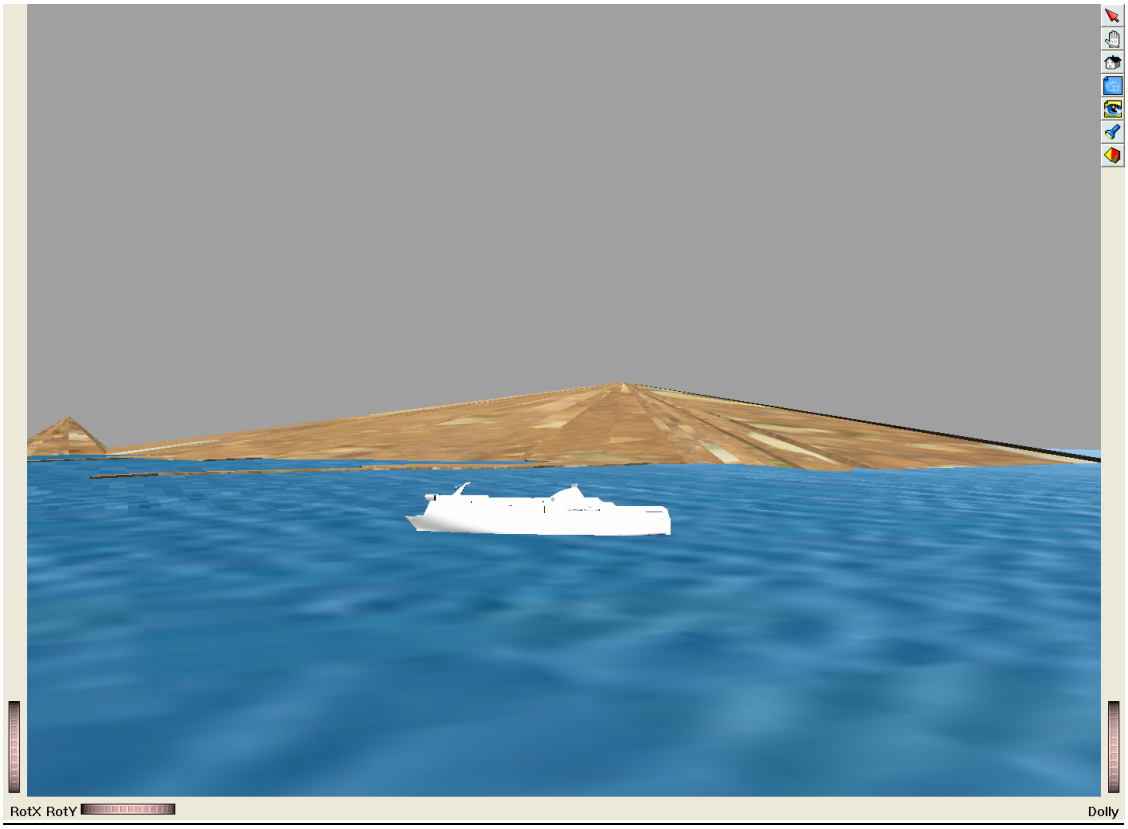

RotX RotY
Dolly

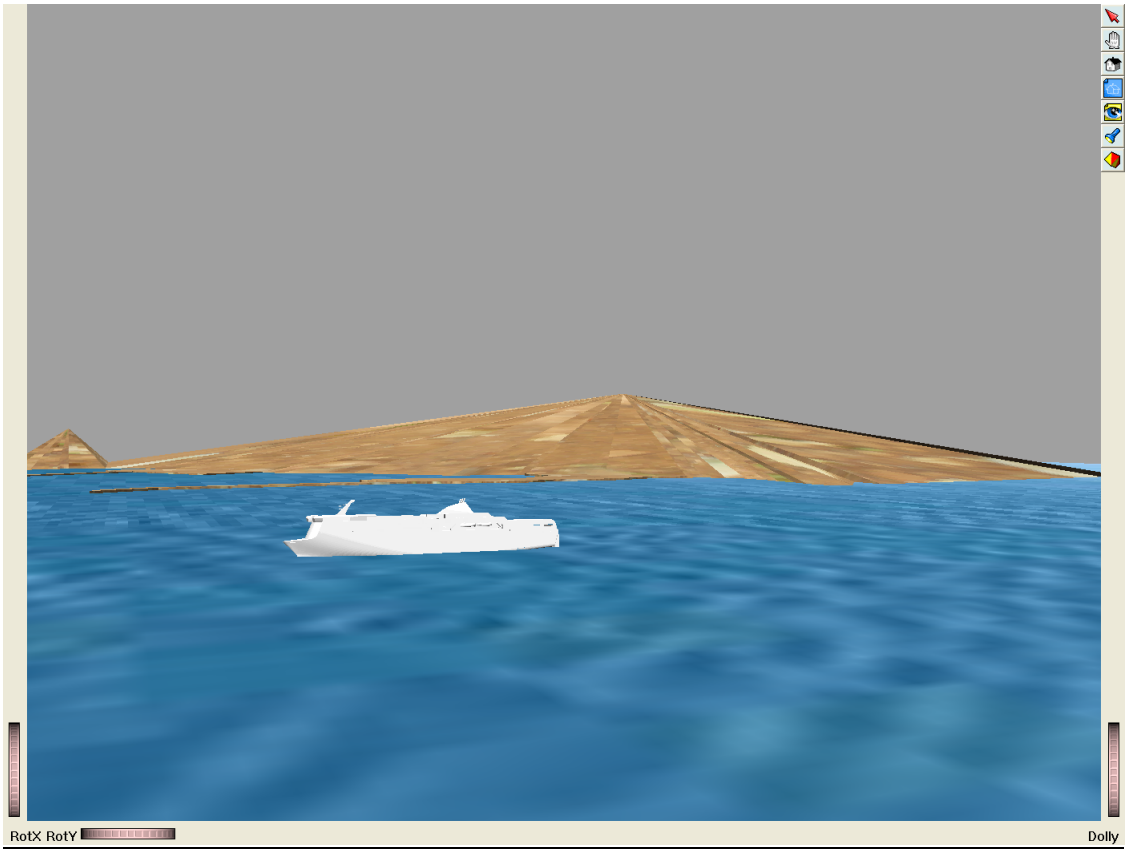

RotX RotY
Dolly

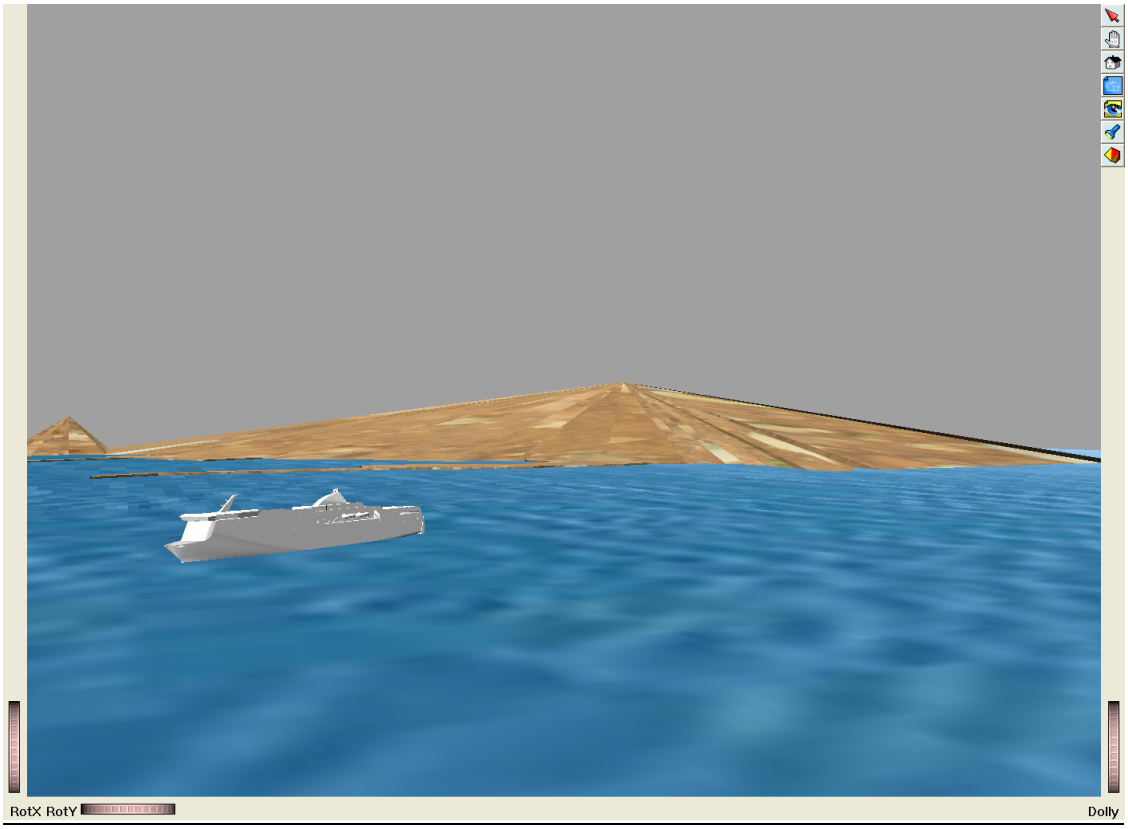

RotX RotY
Dolly

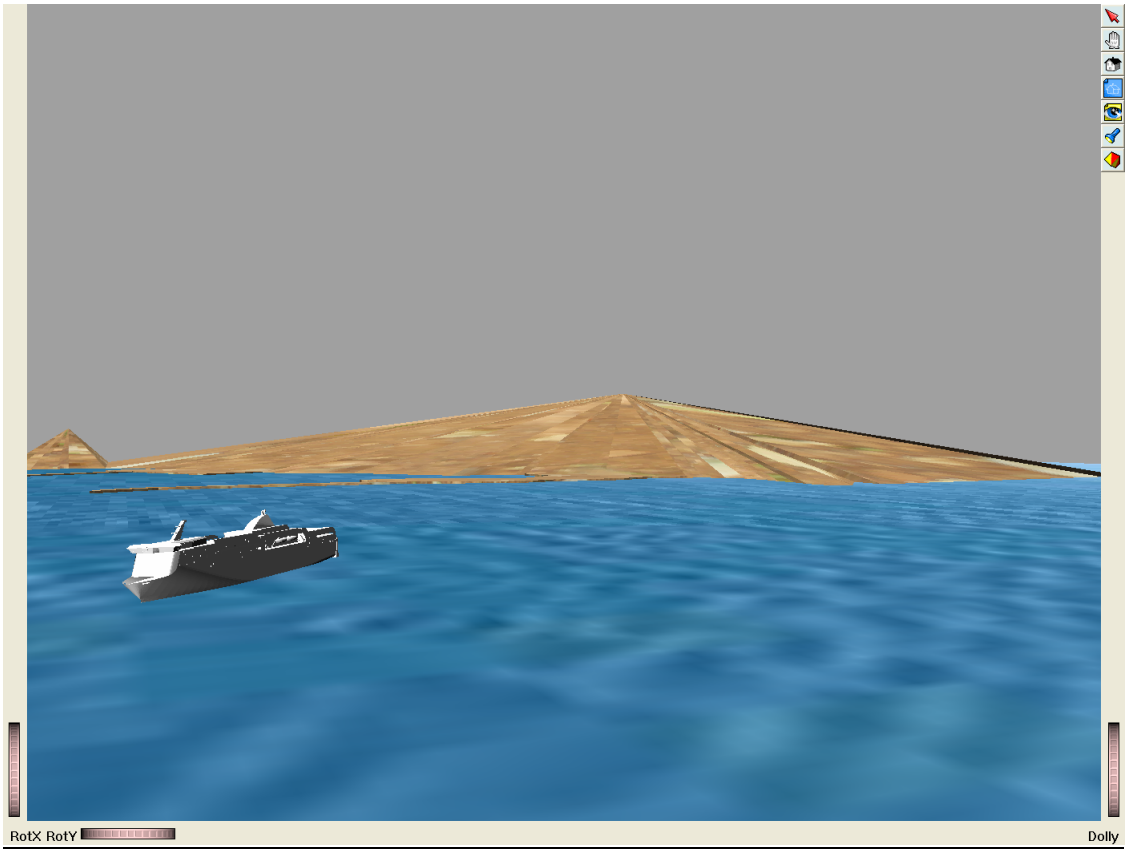

RotX RotY
Dolly

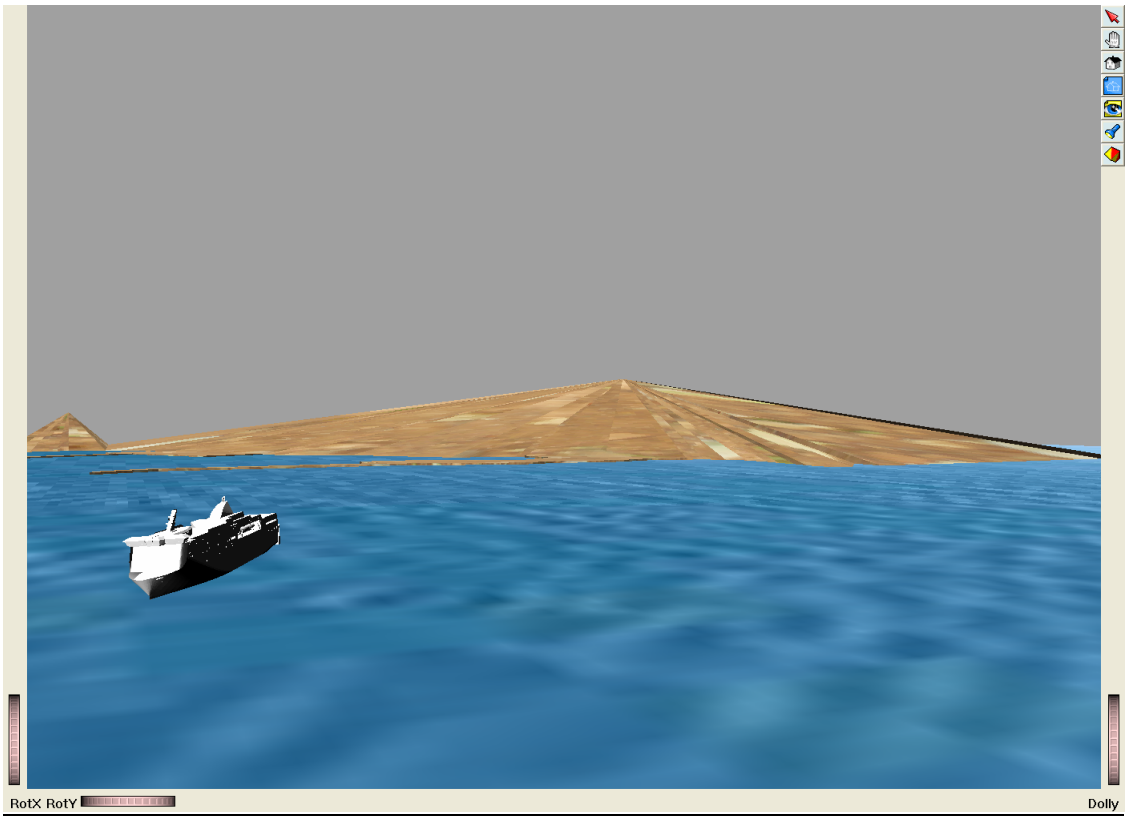

RotX RotY
Dolly

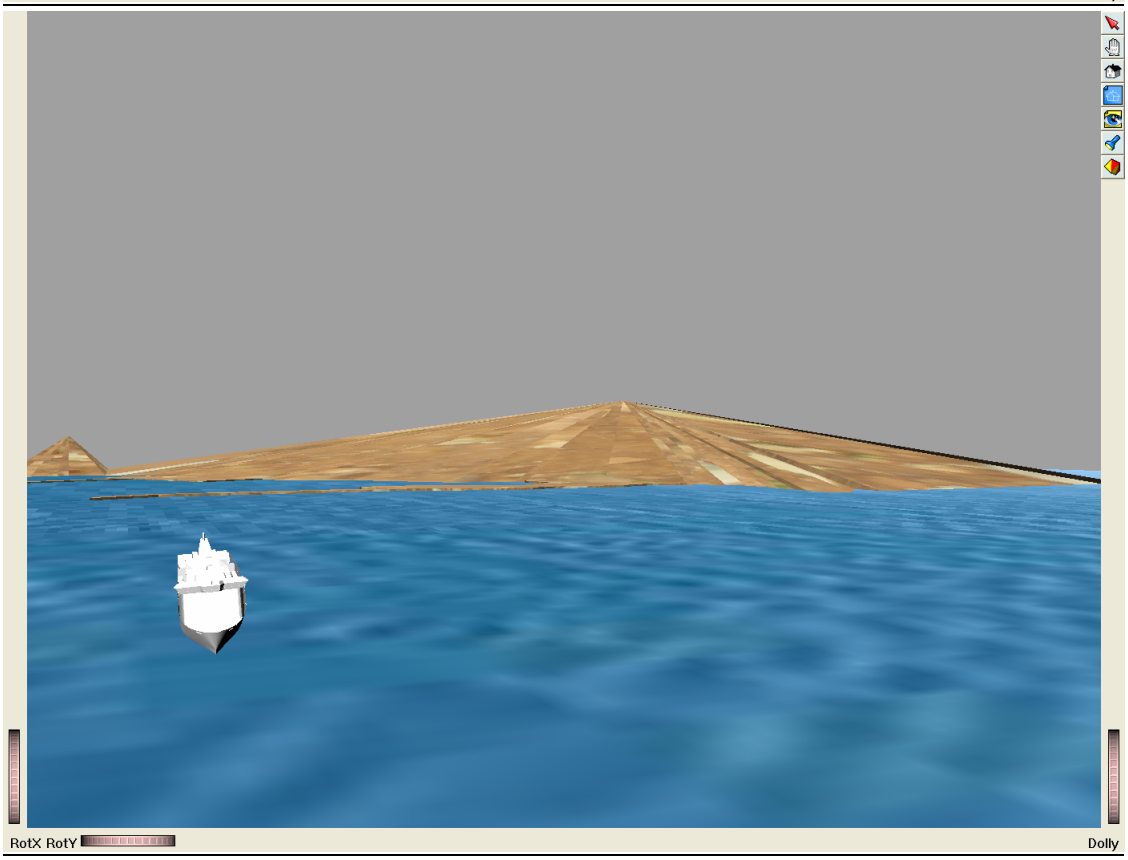

RotX RotY
Dolly

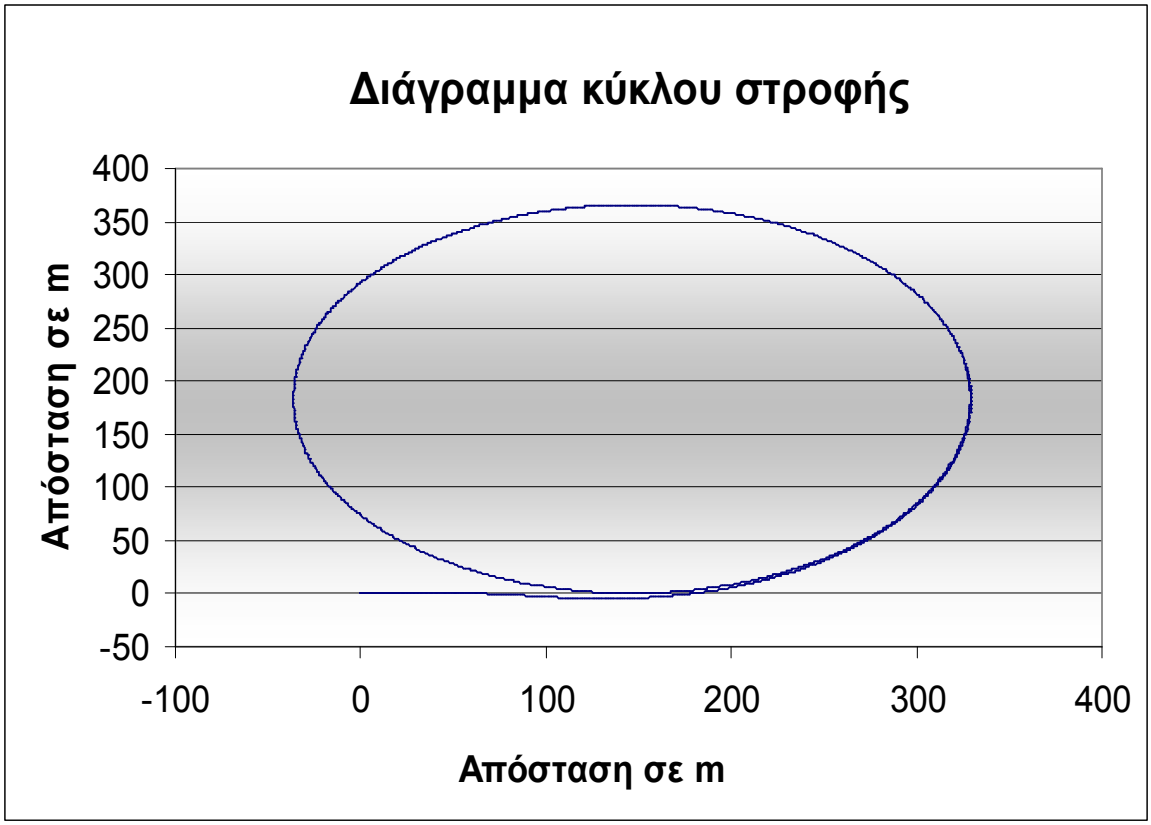

RotX RotY
Dolly

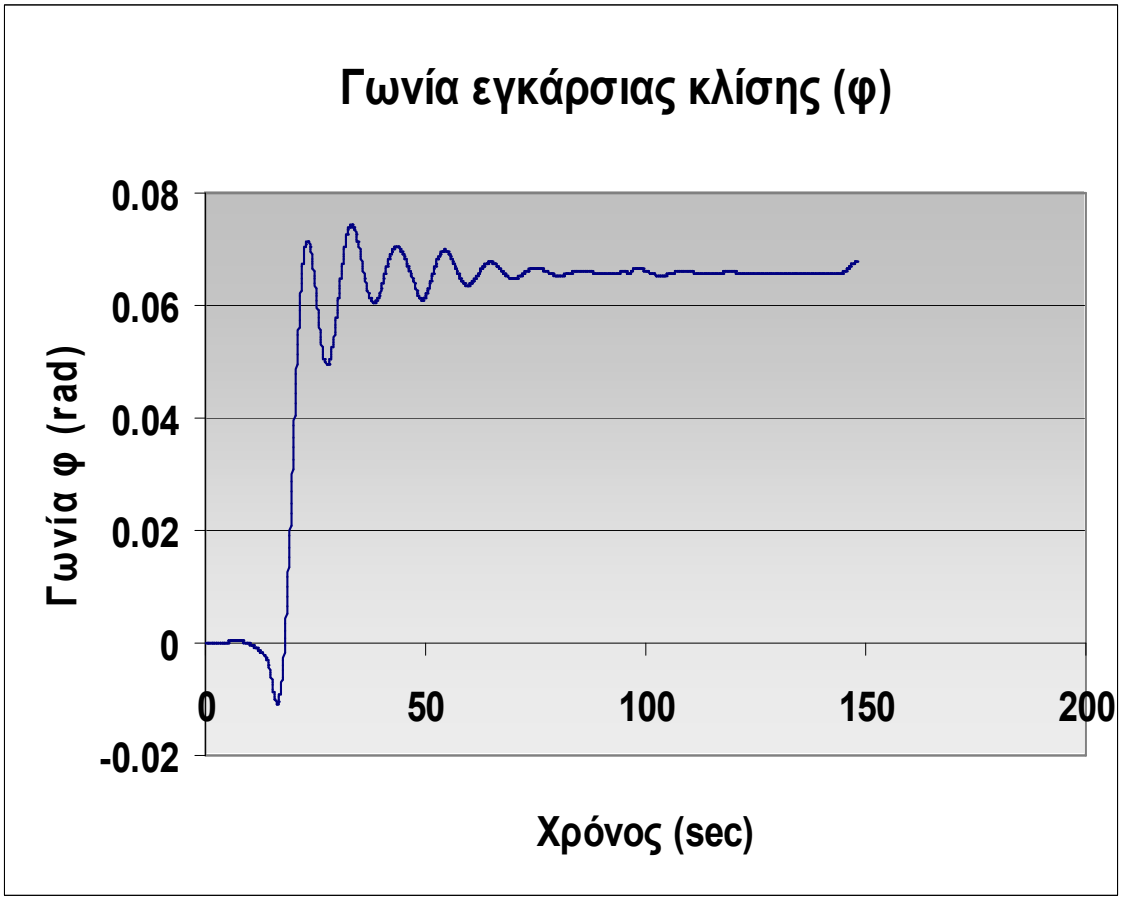

RotX RotY
Dolly


RotX RotY
Dolly


RotX RotY
Dolly


RotX RotY
Dolly

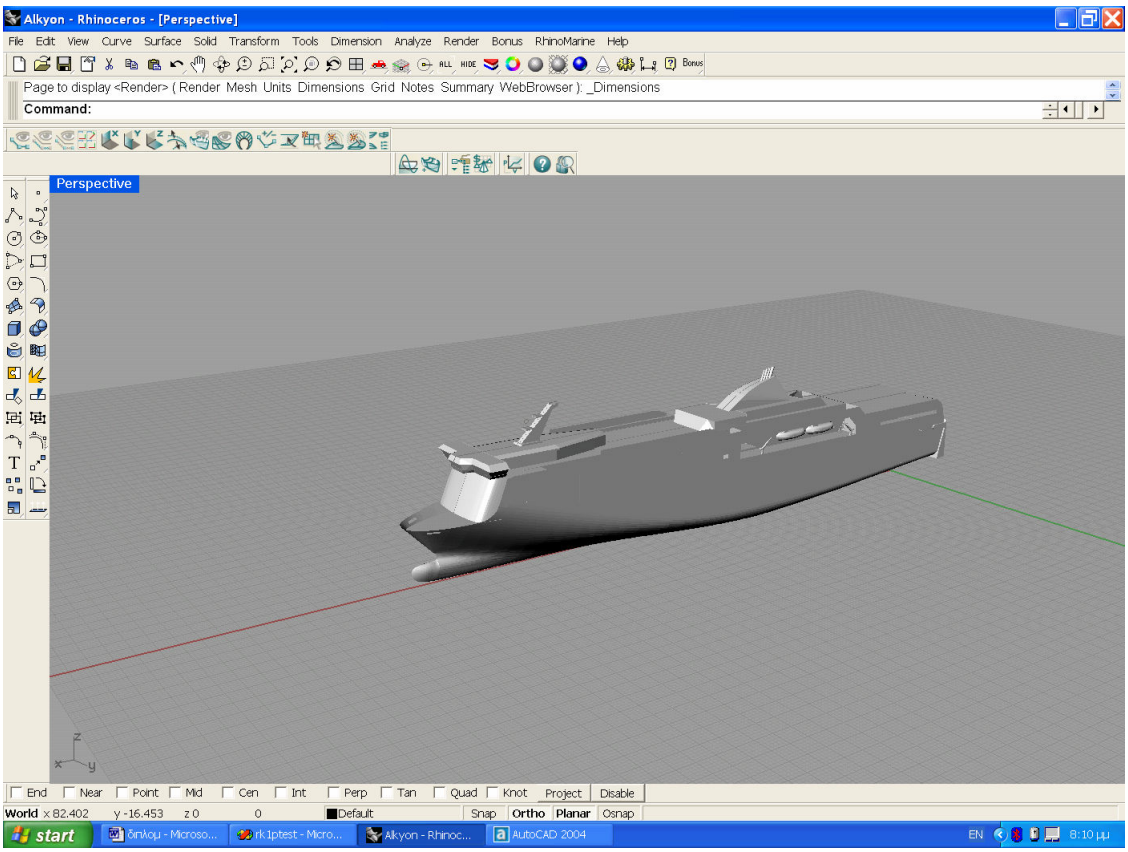

Διάγραμμα κύκλου στροφής
Απόσταση σε m
400
350
300
250
200
150
100
50
0
-50
-100
0
100
200
300
400
Απόσταση σε m


Γωνία εγκάρσιας κλίσης (φ)
Γωνία φ (rad)
0.08
0.06
0.04
0.02
0
-0.02
0
50
100
150
200
Χρόνος (sec)

ΚΕΦΑΛΙΟ 4 ΓΕΩΜΕΤΡΙΚΗ ΑΠΕΙΚΟΝΙΣΗ

4.1 ΤΟ ΕΠΙΒΑΤΗΓΟ/ΟΧΗΜΑΤΑΓΩΓΟ ΑΛΚΥΩΝ.

Η τρισδιάστατη γεωμετρική απεικόνιση του επιβατηγού/οχηματαγωγού Αλκυών προϋπήρχε από προηγούμενη διπλωματική εργασία στην μορφή αρχείου του προγράμματος 3D Studio MAX. Συνεπώς, η πρώτη ενέργειά μας είναι να εξάγουμε αυτή την γεωμετρία σε μορφή που υποστηρίζεται από το πρόγραμμα Rhinoceros 3.0. Αυτό γίνεται με την επιλογή του πλοίου στο πρόγραμμα 3D Studio MAX και την εκτέλεση της εντολής export σε μορφή αρχείου με κατάληξη .3ds που αναγνωρίζει το πρόγραμμα Rhinoceros 3.0.Έτσι εισάγουμε την γεωμετρία του πλοίου στο πρόγραμμα Rhinoceros 3.0 όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 5).



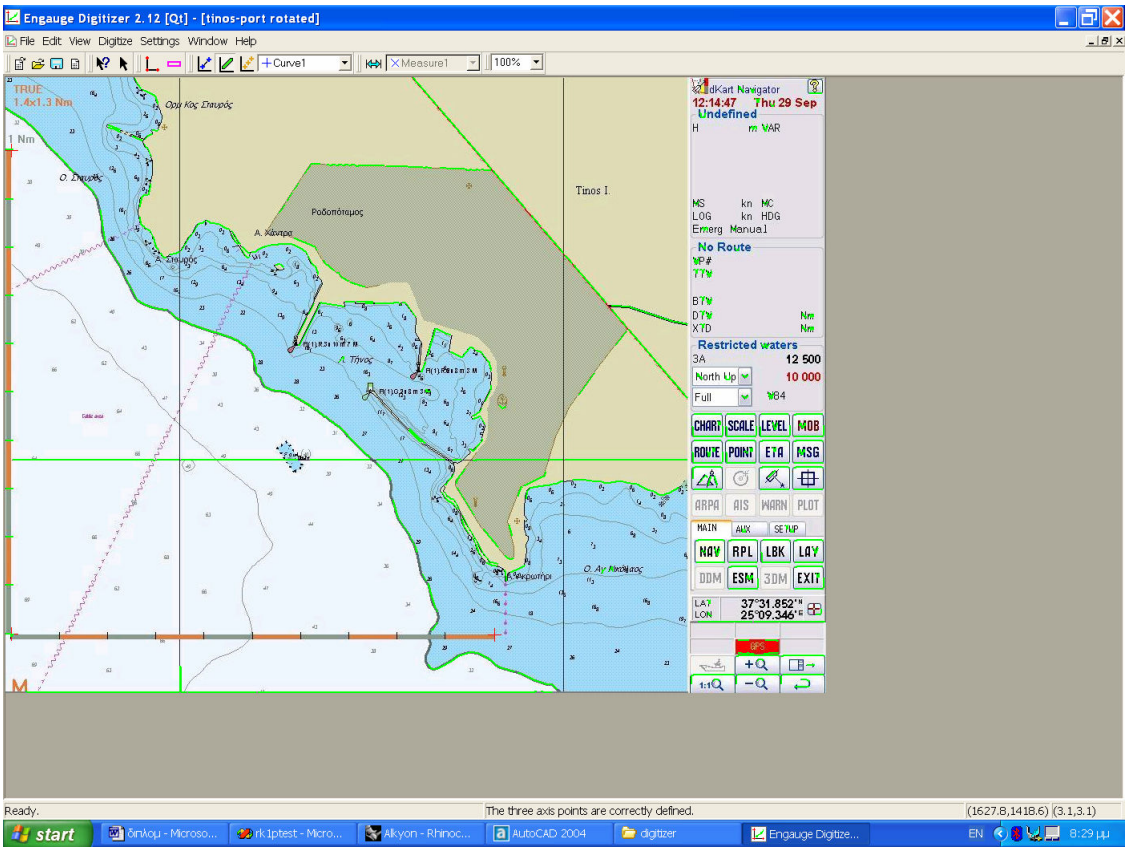
Εικόνα 5. Το πλοίο Αλκυών στο περιβάλλον Rhinoceros v.3.0

Στην συνέχεια, επιλέγουμε το πλοίο και εκτελούμε την εντολή file->export selected. Επιλέγουμε κατάληξη αρχείου .wrl που αναγνωρίζει η βιβλιοθήκη Coin3d Open Inventor.

4.2 Η ΘΑΛΑΣΣΑ ΚΑΙ ΤΑ ΛΙΜΑΝΙΑ

Λογικό είναι οτι πιο δύσκολη είναι η τρισδιάστατη απεικόνιση των λιμανιών παρά της θάλασσας η οποία στην περίπτωσή μας απουσία κυματισμών είναι ένα επίπεδο.
Για την δημιουργία της τρισδιάστατης απεικόνισης της θάλασσας και των λιμανιών ακολουθείται η εξής διαδικασία:

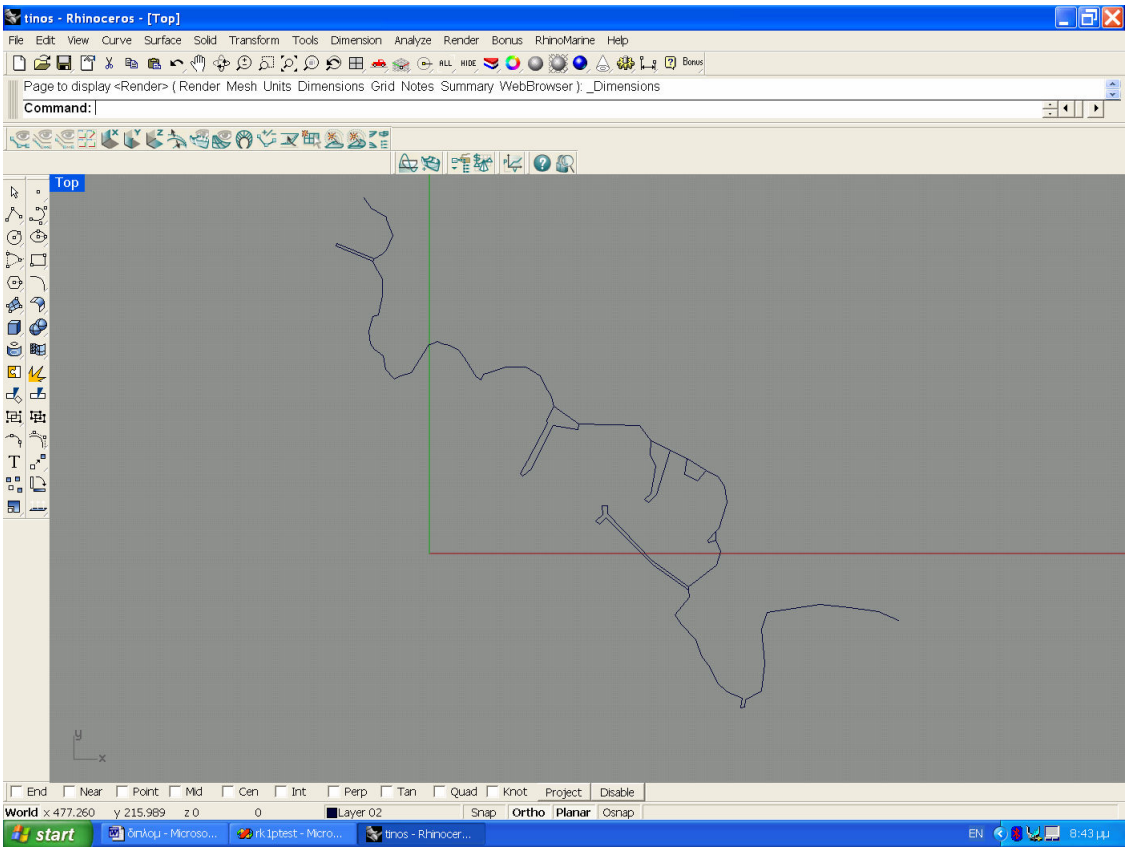
Για την δημιουργία της γεωμετρίας των λιμανιών είναι απαραίτητο να γνωρίζουμε τις συντεταγμένες της ακτογραμμής που ορίζει το λιμάνι. Αυτό επιτυγχάνεται με την χρήση του προγράμματος digitizer. Στο πρόγραμμα αυτό, μπορούμε να εισάγουμε την φωτογραφία ενός χάρτη και να ορίσουμε τους άξονες x και y με την εντολή Digitize – Axis Point.Στην συνέχεια, ορίζουμε και την τιμή που έχει μία απόσταση πάνω στον χάρτη στην πραγματικότητα. Στην ουσία δίνουμε τον ορισμό της κλίμακας του χάρτη στο πρόγραμμα και έτσι το digitizer “ψηφιοποιεί” την εικόνα και μπορεί να υπολογίσει τις συντεταγμένες οποιουδήποτε σημείου πάνω στον χάρτη. Το περιβάλλον εργασίας του digitizer φαίνεται στην παρακάτω εικόνα για το λιμάνι της Τήνου (Εικόνα 6).



Εικόνα 6 .Χάρτης Τήνου στο πρόγραμμα Digitizer.

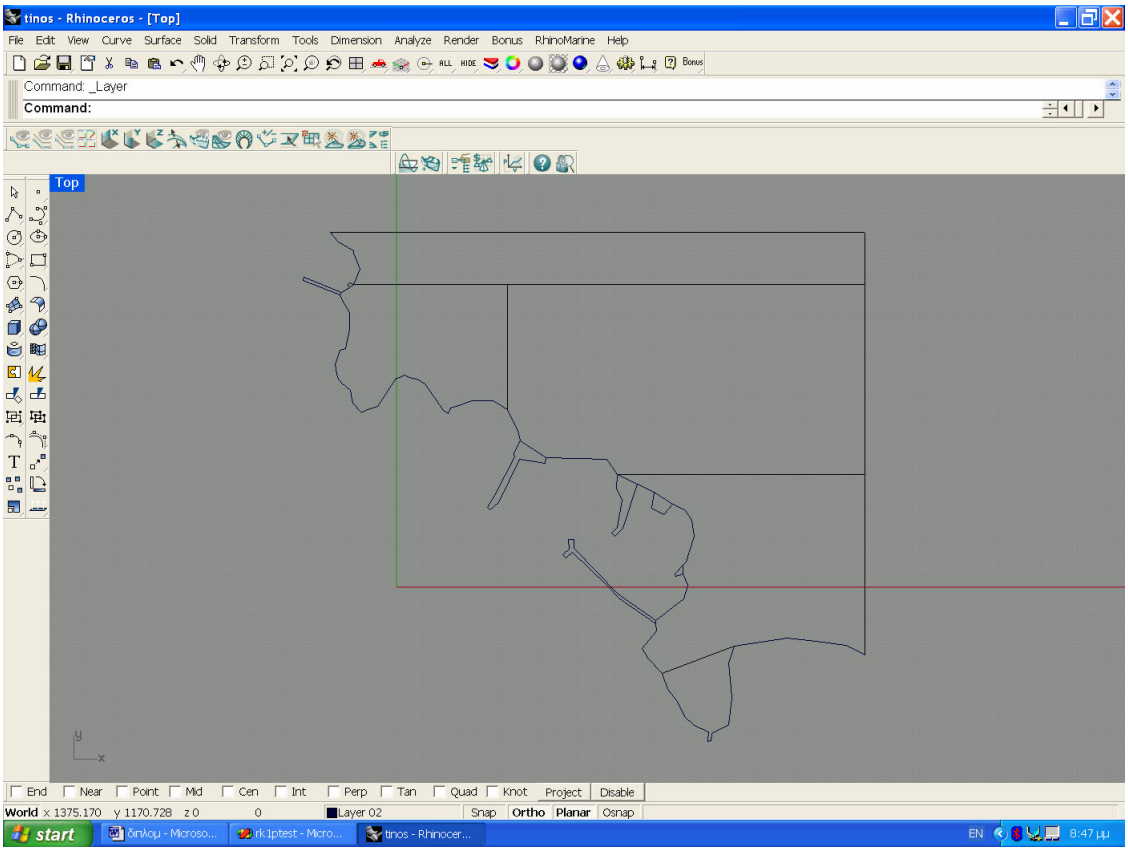
Στην συνέχεια, επιλέγουμε την εντολή Digitize -> Curve Points επιλέγουμε τα σημεία της ακτογραμμής και το πρόγραμμα τυπώνει τις συντεταγμένες τους σε έναν πίνακα. Με απλές εντολές αποκοπής και επικόλλησης μεταφέρουμε τις συντεταγμένες σε ένα αρχείο notepad.

Εισάγουμε τώρα τις συντεταγμένες της ακτογραμμής (της Τήνου στο συγκεκριμένο παράδειγμα) στο πρόγραμμα Rhinoceros 3.0 με την εντολή Import->points file (txt). Ακολούθως, με την εντολή Curve->Polyline δημιουργούμε την καμπύλη της ακτογραμμής όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 7).



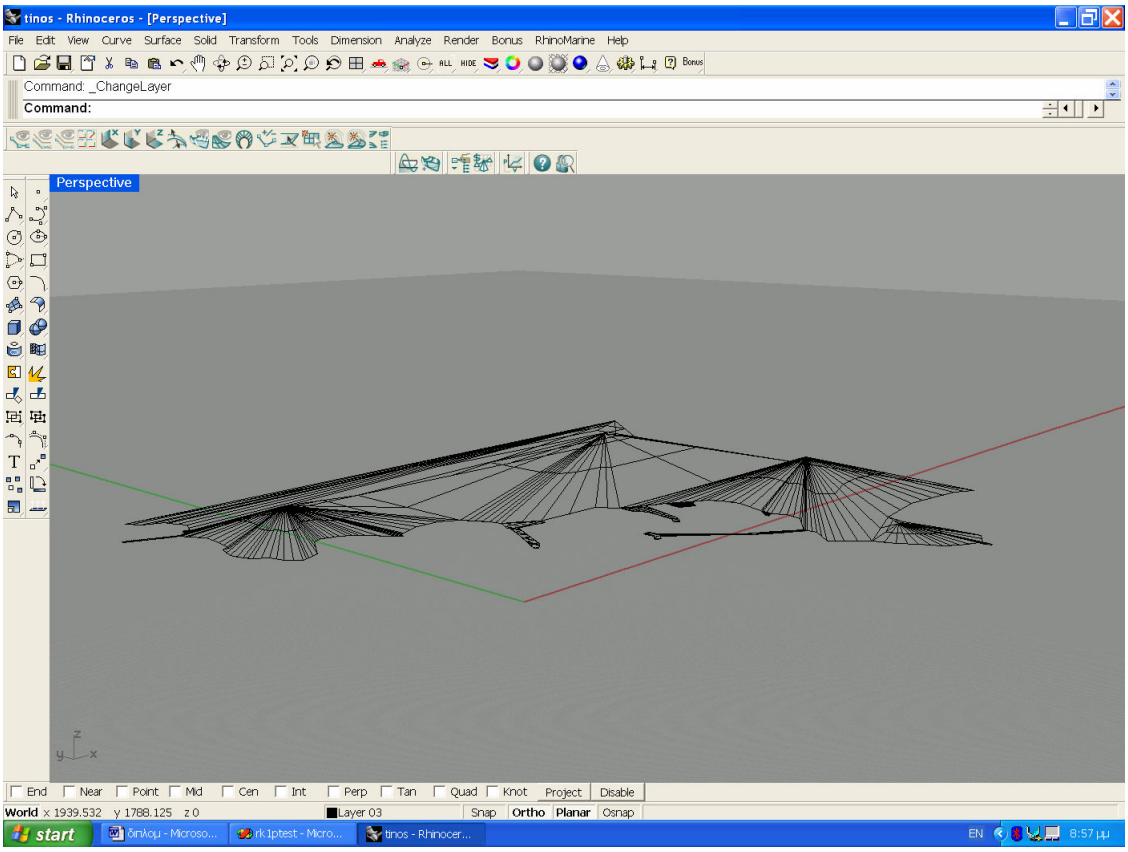
Εικόνα 7. Περίγραμμα λιμανίου Τήνου.

Με βάση την καμπύλη της ακτογραμμής φέρνουμε βοηθητικές γραμμές οι οποίες θα χρησιμοποιηθούν κατά την δημιουργία του τρισδιάστατου μοντέλου του λιμανιού. Οι επιπλέον γραμμές παρουσιάζονται στην παρακάτω εικόνα (Εικόνα 8)



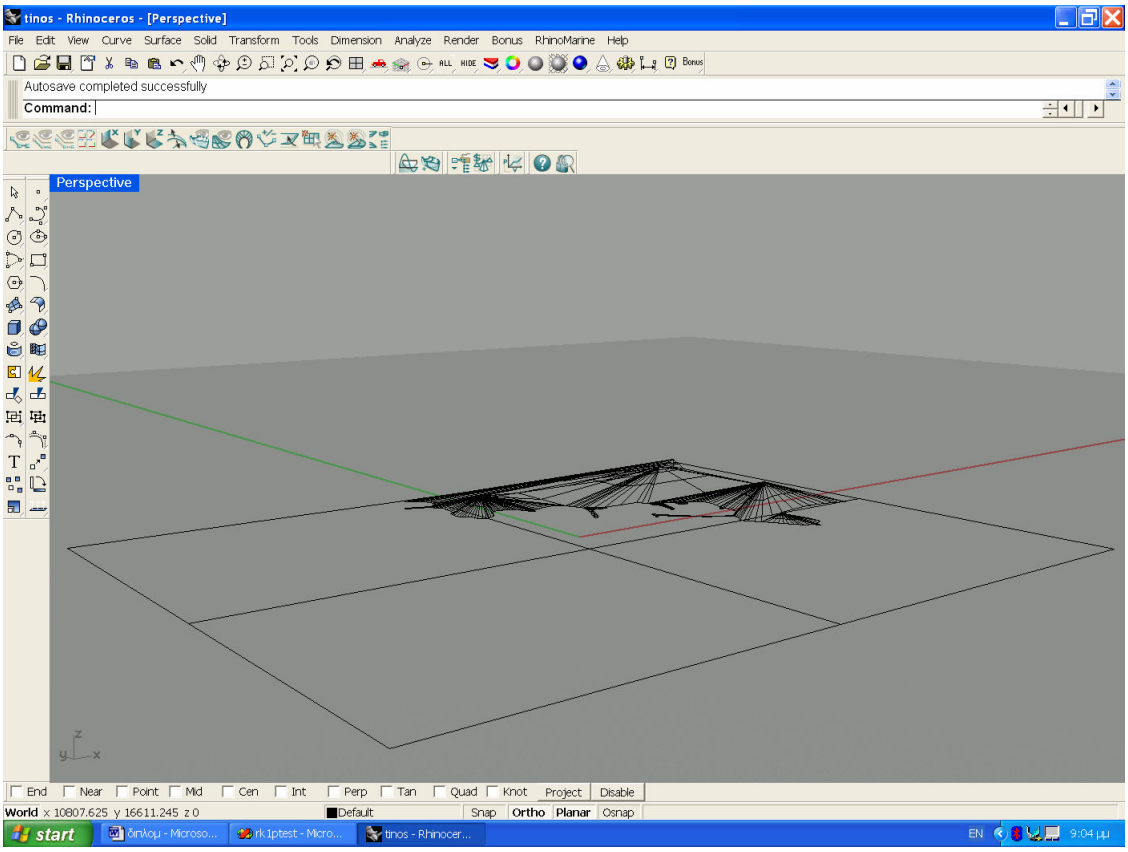
Εικόνα 8. Περίγραμμα λιμανιού Τήνου με βοηθητικές ευθείες.

Με την εντολή Surface->Extrude curve ->to point δημιουργούμε τους λόφους του νησιού ώστε να αποκτήσει μια τρισδιάστατη υπόσταση (το ύψος και η θέση των λόφων δεν είναι ακριβής καθώς δεν μας αφορούν).Στις περιοχές όπου υπάρχουν μόλοι έχει χρησιμοποιηθεί η εντολή Surface->Extrude curve ->straight ώστε να δημιουργείται ένα τρισδιάστατο στερεό καθ' ύψος του περιγράμματος του μόλου. Τελικά, έχουμε δημιουργήσει ένα τρισδιάστατο μοντέλο του λιμανιού του νησιού (Εικόνα 9).



Εικόνα 9. Τρισδιάστατο μοντέλο λιμανιού Τήνου.

Η απεικόνιση της θάλασσας στην περίπτωσή μας είναι πολύ απλή αφού δεν υπάρχουν κυματισμοί που να επηρεάζουν την ελεύθερη επιφάνειά της.
Συνεπώς, η θάλασσα αποτελείται από ένα επίπεδο το οποίο δημιουργούμε στο πρόγραμμα Rhinoceros 3.0 με την εντολή plane και ορίζουμε τα ακραία σημεία της. Αυτό γίνεται στο ίδιο αρχείο που βρίσκεται και η τρισδιάστατη απεικόνιση του λιμανιού ώστε να έχουμε και τις δύο γεωμετρίες στο τελικό μας πρόγραμμα. Είναι προφανές οτι μπορούν να γίνουν διάφορες παραλλαγές.(μόνο θάλασσα, θάλασσα και 1 λιμάνι, θάλασσα και 2 λιμάνια κ.α.). Το αποτέλεσμα φαίνεται στην εικόνα που ακολουθεί (Εικόνα 10).

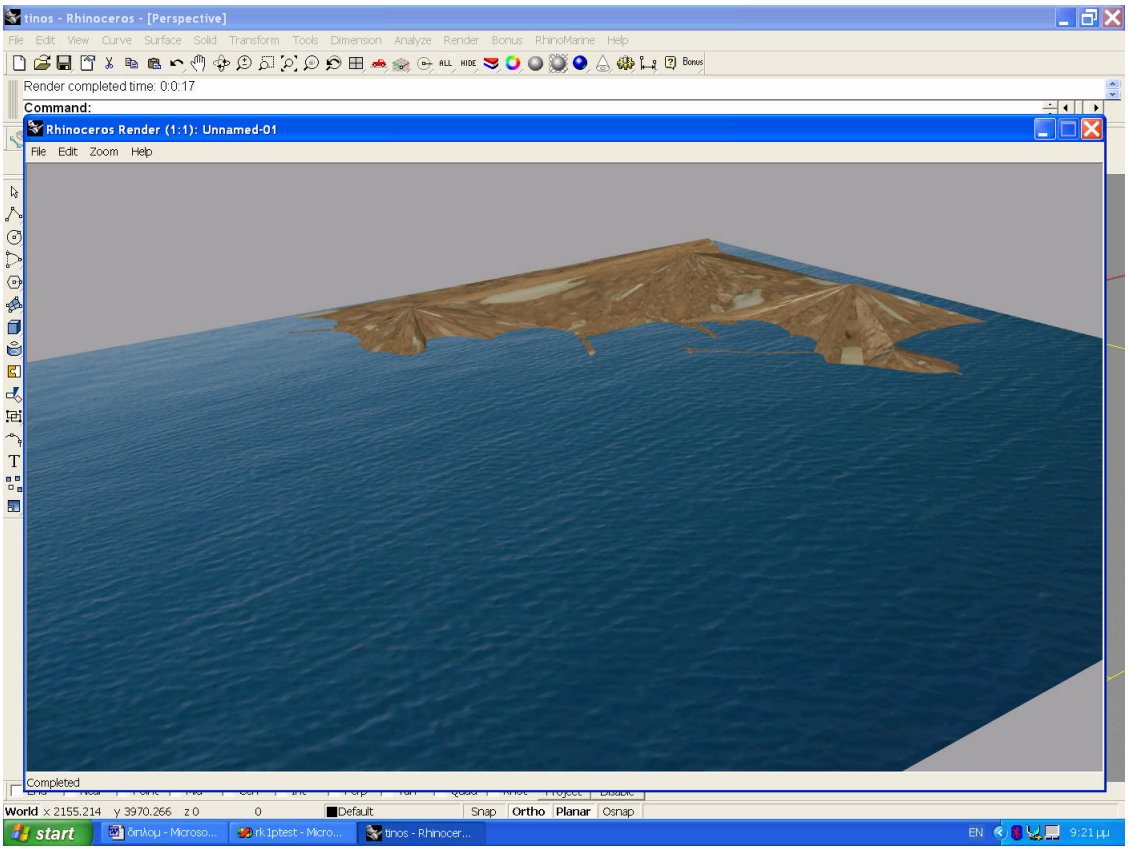


Εικόνα 10. Τρισδιάστατο μοντέλο λιμανιού Τήνου με θάλασσα.

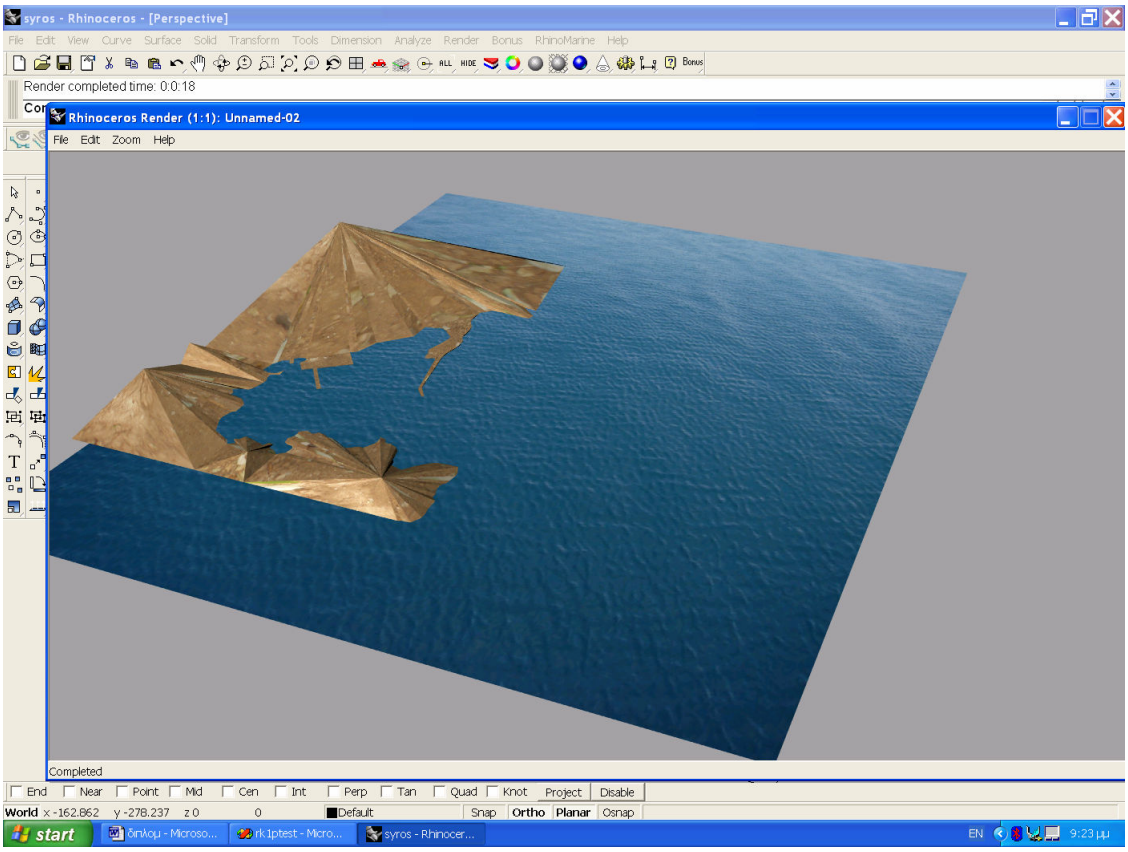
4.3 TEXTURES

Για να είναι η επιφάνεια των αντικειμένων ρεαλιστική κατά την γραφική τους απεικόνιση χρησιμοποιούμε τα textures. Πρόκειται στην ουσία για φωτογραφίες οι οποίες επικολλώνται πάνω στην επιφάνεια δίνοντάς της μια πιο πραγματική όψη. Θέλει όμως προσοχή κατά την επιλογή των φωτογραφιών προς επικόλληση. Μια φωτογραφία μεγάλη σε δεδομένα (άνω των 500KB) μπορεί να προκαλέσει προβλήματα στον επεξεργαστή κατά την διάρκεια εκτέλεσης του προγράμματος προσομοίωσης και το αποτέλεσμα να υστερεί σημαντικά από άποψη χρόνου σε σχέση με την πραγματικότητα.
Επίσης η προσθήκη πολλών φωτογραφιών σε πολλές γεωμετρίες προκαλεί και αυτή προβλήματα στον επεξεργαστή του υπολογιστή. Γενικώς,η χρήση των textures πρέπει να γίνεται με μέτρο και προσοχή.

Στην περίπτωσή μας επιλέξαμε μία φωτογραφία θάλασσας για επικόλληση πάνω στην επιφάνεια της θάλασσας και μία φωτογραφία του εδάφους της γης για επικόλληση πάνω στην επιφάνεια των νησιών. Αυτό γίνεται με την εντολή Odject properties->Material->basic->texture. Το τελικό αποτέλεσμα και αυτό που τελικά εισέρχεται στο πρόγραμμα που εκτελεί την προσομοίωση του πλοίου Αλκυών παρουσιάζεται στις εικόνες που ακολουθούν για το λιμάνι της Τήνου και της Σύρου (Εικόνα 11, Εικόνα 12).



Εικόνα 11. Τελική Τρισδιάστατη Απεικόνιση λιμανιού Τήνου με εφαρμογή των textures.



Εικόνα 12. Τελική Τρισδιάστατη Απεικόνιση λιμανιού Σύρου με εφαρμογή των textures.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Το πιο σημαντικό πλεονέκτημα που διαθέτει η παρούσα διπλωματική εργασία (που ήταν και ο βασικός της στόχος) είναι η δυνατότητα της αλληλεπίδρασης του χρήστη με το εικονικό περιβάλλον σε πραγματικό χρόνο. Για την επίτευξη αυτού του στόχου χρησιμοποιήθηκε η βιβλιοθήκη Coin3d Open Inventor και η γλώσσα C++. Στη αντίθετη περίπτωση, αν επιθυμούσαμε ένα απλό animation χωρίς την επέμβαση του χρήστη σε πραγματικό χρόνο, θα μπορούσαμε να χρησιμοποιήσουμε "έτοιμα" προγράμματα (3d Studio MAX) που δεν απαιτούν υψηλές γνώσεις προγραμματισμού.

Η επέμβαση του χρήστη σε πραγματικό χρόνο στο περιβάλλον που εκτελεί την προσομοίωση δίνει μεγάλες δυνατότητες να προσομοιωθούν πολλές και πολύπλοκες καταστάσεις. Μεταβάλλοντας τις στροφές της έλικας και την γωνία του πηδαλίου ο χρήστης κυβερνά πλέον το πλοίο. Συνεπώς, μπορεί να εκτελεστεί ένα ευρύ φάσμα εφαρμογών και να μελετήσει τις κινήσεις του πλοίου.

Αρχικά,μπορούν να εκτελεστούν οι κλασικές δοκιμές της ελικτικότητας και της δυναμικής ευστάθειας του πλοίου. Δοκιμές zig – zag, κύκλος στροφής, ο ευθύς και ο αντίστροφος σπειροειδής ελιγμός, η δοκιμή επαναφοράς και στην ουσία, ότι άλλο μπορεί να θελήσει ο χρήστης να δοκιμάσει στο πλοίο που κυβερνά.

Ακόμη, υπάρχουν στο πρόγραμμα και οι τρισδιάστατες απεικονίσεις των λιμανιών της Σύρου και της Τήνου. Έτσι ο χρήστης-κυβερνήτης μπορεί να δοκιμάσει την προσέγγιση των λιμανιών και να βγάλει σημαντικά συμπεράσματα για την δυνατότητα του πλοίου να εκτελέσει ελιγμούς σε περιορισμένα νερά.

Οι επιλογές που δίνονται από το παράθυρο παρακολούθησης (περιστροφές κατά τους άξονες x και y και zoom) μας επιτρέπουν να παρατηρήσουμε το πλοίο από οποιαδήποτε γωνία και απόσταση (ακόμη και κάτω από την ίσαλο γραμμή). Αυτό είναι ένα αρκετά χρήσιμο εργαλείο διότι μπορούμε να κινήσουμε την κάμερα ώστε να "συγκεντρώνεται" στην πτυχή της κίνησης που μας ενδιαφέρει περισσότερο (π.χ. κατά την διάρκεια του κύκλου στροφής η κίνηση roll).

Όπως έχουμε αναφέρει έχει γίνει προσπάθεια να είναι ο κώδικας ευανάγνωστος και ευκατανόητος για άλλους προγραμματιστές ή απλούς χρήστες. Έτσι η εισαγωγή στο πρόγραμμα ενός άλλου πλοίου είναι αρκετά απλή καθώς και η εισαγωγή διαφορετικών λιμανιών (γενικώς γεωμετριών) που έχουμε ανάπτυξη σε ένα CAD πρόγραμμα.

Γίνεται αντιληπτό λοιπόν οτι ο κώδικας μπορεί να χρησιμοποιηθεί για πολλά πλοία και λιμάνια με απλές τροποποιήσεις του που δεν απαιτούν πολλές γνώσεις προγραμματισμού. Το γεγονός αυτό είναι ιδιαίτερα σημαντικό κατά την διάρκεια της μελέτης ενός πλοίου προς κατασκευή ή ενός πλοίου που δεν υπάρχουν στοιχεία δοκιμών των ελικτικών δυνατοτήτων του. Ο μελετητής μπορεί να εισάγει το υπό μελέτη πλοίο στο πρόγραμμα, να εκτελέσει τις δοκιμές ελικτικότητας και δυναμικής ευστάθειας που επιθυμεί και να βγάλει χρήσιμα συμπεράσματα.

Το πρόγραμμα που αναπτύξαμε έχει και διδακτικό χαρακτήρα. Για έναν σπουδαστή που μελετάει τις έννοιες της ελικτικότιτας, της δυναμικής ευστάθειας και της πηδαλιουχίας η παρατήρηση της τρισδιάστατης γραφικής αναπαράστασης των αποκρίσεων του πλοίου με αυξομείωση των στροφών της έλικας και εκτροπή του πηδαλίου αποτελεί ένα σημαντικό μάθημα για την καλύτερη κατανόηση των εννοιών αυτών.

Όλα τα παραπάνω πλεονεκτήματα του προγράμματος δεν σημαίνουν οτι δεν είναι επιθυμητή η επέκταση του κώδικα ώστε το πρόγραμμα να αποκτήσει κι άλλες δυνατότητες.

Έχει ήδη αναφερθεί ότι στην συγκεκριμένη εφαρμογή δεν εισέρχονται κυματισμοί. Παρ' όλα αυτά είναι δυνατή μια τροποποίηση του κώδικα ώστε να υπάρχει και η επίδραση των κυματισμών. Φυσικά η αναπαράσταση των κυματισμών γραφικά και η αλληλεπίδραση με το πλοίο αποτελεί σαφώς πιο δύσκολο εγχείρημα όμως η «μαθηματική, υπολογιστική» λύση του προβλήματος, χωρίς τις γραφικές απεικονίσεις, είναι αρκετά απλή από άποψη προγραμματισμού.

Η προσθήκη της επίδρασης των δυνάμεων του ανέμου είναι πιο απλή να προσομοιωθεί μιας και δεν απατιούνται μετατροπές στο γραφικό κομμάτι του κώδικα. Έτσι με απλές μετατροπές στο υπολογιστικό κομμάτι θα μπορούσαμε να συμπεριλάβουμε την επίδραση του ανέμου στην ελικτικότητα του πλοίου.

Μια ακόμη ελκυστική προσθήκη θα ήταν η δυνατότητα επιλογής διαφόρων καμερών, με ένα πλήκτρο, κατά την διάρκεια της παρακολούθησης της προσομείωσης. Αυτό σημαίνει οτι θα πρέπει να "θυσιάσουμε" την κάμερα που μας επιτρέπει περιστροφές κατά τους άξονες x και y και zoom. Θα εισάγουμε άλλες κάμερες σε διάφορες σημαντικές θέσεις (π.χ. μια στην γέφυρα του πλοίου, μια στο πλάι, μια από πίσω, μια πανοραμική κ.α.) και με ένα πλήκτρο θα επιλέγουμε μέσα από ποια κάμερα θέλουμε να παρατηρήσουμε την προσομοίωση.

Αξίζει να σημειωθεί οτι η βιβλιοθήκη Coin3d Open Inventor έχει πολλές δυνατότητες για ένα μεγάλο εύρος τρισδιάστατων γραφικών απεικονίσεων και εφαρμογών. Υπάρχει η δυνατότητα να εισαχθούν πολλά αντικείμενα,να οριστούν πολλές αλληλεπιδράσεις του χρήστη σε πραγματικό χρόνο, πολλές κάμερες κ.α. Το γεγονός οτι είναι γραμμένη σε γλώσσα C++ και προφανώς επικοινωνεί με αυτήν μας λύνει τα χέρια από υπολογιστικής άποψης. Το

αντίτιμο είναι οτι ο προγραμματιστής πρέπει να είναι αρκετά εξοικειωμένος με την γλώσσα C++.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] ΣΧΕΔΙΑΣΗ ΠΛΟΙΩΝ ΓΙΑ ΕΛΙΚΤΙΚΟΤΗΤΑ ΚΑΙ ΔΥΝΑΜΙΚΗ ΕΥΣΤΑΘΕΙΑ, Κ. ΣΠΥΡΟΥ ΕΜΠ.

[2] ΥΔΡΟΔΥΝΑΜΙΚΗ ΚΑΙ ΔΥΝΑΜΙΚΗ ΘΑΛΑΣΣΙΩΝ ΣΥΣΤΗΜΑΤΩΝ, Γ. ΑΘΑΝΑΣΟΥΛΗΣ Θ.ΛΟΥΚΑΚΗΣ

[3] C++ HOW TO PROGRAM Second Edition, DEITEL & DEITEL

[4] THE INVENTOR MENTOR: Programming Object Oriented 3d Graphics with Open Inventor™

[5] Coind3d Documentation version 2.3.0

ΠΑΡΑΡΤΗΜΑ

Στο εδάφιο αυτό παρατίθεται ο κώδικας που εκτελεί την προσομοίωση της κίνησης του πλοίου Αλκυών σε ήρεμο (παρουσία του λιμανιού της Σύρου) και την αλληλεπίδραση του χρήστη με το περιβάλλον. Όπως αναφέρθηκε στο κεφάλαιο ΙΙΙ ο συνολικός κώδικας αποτελείται από τρία αρχεία. Το header file shipspescs.h, το functions.cpp και το simulation.cpp. Έχει γίνει προσπάθεια ο κώδικας να είναι όσο το δυνατόν πιο "τακτοποιημένος" ώστε να είναι ευανάγνωστος σε κάποιον άλλο προγραμματιστή και ευκατανόητος σε κάποιον αναγνώστη όχι και τόσο έμπειρο στην γλώσσα C++.

Κάτω από το αρχείο shipspecs.h παρατίθεται και ο πίνακας που περιέχει τις τιμές (σταθερές) όλων των απαραίτητων γεωμετρικών και υδροδυναμικών χαρακτηριστικών του πλοίου Αλκυών που βρίσκονται στο αρχείο shipspecs.h και χρησιμοποιούνται από το πρόγραμμα προσομείωσης. Με αυτόν τον τρόπο είναι αρκετά απλό και για έναν χρήστη με ελάχιστη εμπειρία στον προγραμματισμό να προσδιορίσει σε ποια θέση εισέρχονται οι νέες τιμές στην περίπτωση που θέλει να εκτελέσει την προσομοίωση ενός άλλου πλοίου με διαφορετικά χαρακτηριστικά.

Αρχείο shipspecs.h

```
#define SIZE 50

//  mass properties //

double sm = 15993500 ;
double xG = -9.26;
double zG = 4.69;
double sIx = 1550279750 ;
double sIz = 35705788000;

// propeller properties //

double aan =3.398605;
double D = 5;
double pitch = 5.97;
double BLAR = 0.75 ;
double xP =-81.5 ;


// rudder properties //

double RAREA =22;
double alamda = 2.27 ;
double HR = 5;
double xR0 = -82.6;
double zR0 = 3.7 ;

// hydrodynamic coefficients 1 //

double YYvdot = -13445415 ;
double YYrdot = -79951391 ;
double YXvr = -6085731 ;
double YXudot =  -1344541.5 ;
double YNvdot = -10049649.4 ;
double YNrdot  = -23557572000 ;

// hydrodynamic coefficients 2 //

double YYv  = -133935.4;
double YYr  = 5729382.3;
double YYvv = -329472.08;
double YYvr =  -24018970.7;
double YYrr = -792814380;
```

```
// hydrodynamic coefficients 3A //

double YNr = -560810620 ;
double YNv = -6210522 ;
double YNrr = -121608192000 ;
double YNrrv = -17179984000 ;
double YNvvr = -3442848000.00 ;
double YNphi  = -714193.5 ;
double YNvphi  =  -10657255.9;

// hydrodynamic coefficients 3B //

double YNrphi = 1323513000 ;

// hydrodynamic coefficients 4 //

double YKphi = -546667620;
double YKp = -172636609 ;
double zY = 2.48 ;
double YKpdot  = -146875828;

// wake fr //

double wP0 = 0.093 ;

double wR0  = 0.093;

// thrude //

double tP0 = 0.111 ;
double tR0 = 0.111 ;

// paramaters //

double sL  = 172;
double drt  = 6.2;
double Disp = 15993500 ;
double CM  = 0.925;

// threshold //

double a1  = 48318.6;
double a2 =  -6821.121 ;
double a3 = 864.1786 ;
double cc1  = 0.57222 ;
double c2  = -0.3925926;
double c3  = -0.0493827;
```

// wave //

```
double WL = 0.0 ;
double HW = 0.0 ;

double BA1=(sIz-YNrdot)*(sIx-YKpdot);
double BA2=(YYrdot-sm*xG)*(sIx-YKpdot)-zY*YYrdot*sm*zG;
double BA3=(sIz-YNrdot)*sm*zG;
double BB1=(YNvdot-sm*xG)*(sIx-YKpdot);
double BB2=(sm-YYvdot)*(sIx-YKpdot)+sm*zG*(zY*YYvdot-sm*zG);
double BB3=sm*zG*(YNvdot-sm*xG);
double BC1=zY*YYrdot*(sm*xG-YNvdot)-(sIz-YNrdot)*(zY*YYvdot-sm*zG);
double BC2=zY*YYrdot*(YYvdot-sm)-(YYrdot-sm*xG)*(zY*YYvdot-sm*zG) ;
double BC3=(sm-YYvdot)*(sIz-YNrdot)-(sm*xG-YNvdot)*(sm*xG-YYrdot);

double DET=(sm-YYvdot)*(sIz-YNrdot)*(sIx-YKpdot)-(sm*xG-
YYrdot)*(sm*xG-YNvdot)*(sIx-YKpdot)-sm*zG*((sm*xG-
YNvdot)*(zY*YYrdot)-(sIz-YNrdot)*(zY*YYvdot-sm*zG));
```

ΠΑΡΑΜΕΤΡΟΣ	**ΤΙΜΗ**	**ΜΟΝΑΔΕΣ**
YY_{vdot}	-13445415	kgr
YYr_{dot}	-79951391	kgr
YX_{vr}	-6085731	kgr
YX_{udot}	-1344541.5	kgr
YN_{vdot}	-10049649.4	kgr
YN_{rdot}	-23557572000	kgr
YY_{v}	-133935.4	kgr
YY_{r}	5728382.3	kgr
YY_{vv}	-329472.08	kgr
YY_{vr}	-24018970.7	kgr
YY_{rr}	-792814380	kgr
YN_{r}	-560810620	kgr
YN_{v}	-6210522	kgr
YN_{rr}	-1.21608E+11	kgr
YN_{rrv}	-1717998400	kgr
YN_{vvr}	-3442848000	kgr
YN_{phi}	-714193.5	kgr
YN_{vphi}	-10657255.9	kgr
YN_{rphi}	1323513000	kgr
YK_{phi}	-546667620	kgr
YK_{p}	-173636609	kgr
zY	2.48	m
YK_{pdot}	-146875828	kgr

wPO	0.093	
wRO	0.093	
tPO	0.111	
tRO	0.111	
sm	15993500	kgr
xG	-9.26	m
zG	-4.69	m
sIx	1550279750	kgr*m^2
sIz	35705788000	kgr*m^3
aan	3.398605	rps
D	5	m
Pitch	5.97	m
xP	-81.5	m
BLAR	0.75	
RAREA	22	m^2
alamda	2.27	
HR	5	m
xRO	-82.6	m
zRO	3.7	m
sL	172	m
DRT	6.2	m
DISP	15993500	kgr
CM	0.925	
a1	48318.6	
a2	-6821.121	
a3	864.1786	
cc1	0.57222	
c2	-0.3925926	
c3	-0.0493827	
WL	0	
HW	0	

Αρχείο functions.cpp

```
#include <math.h>
#include <shipspecs.h>


// CALCULATION OF SWAY FORCES //

// calculation of sway propeller forces fyp//

double fyp ( double u)
{
int po=1025;
double YPC;
double ansfyp;

if (u>0 && aan<0)

YPC = 0.1;
else
YPC = 0.0;

ansfyp=po*(aan*aan)*(D*D*D*D)*YPC;

return ansfyp;

}

//calculation of the hull sway forces fyh //


double fyh ( double u,double v,double r)
{
double ansfyh;
double abs_v = v, abs_r = r;

if (abs_v < 0)
abs_v = -abs_v;
if (abs_r < 0)
abs_r = -abs_r;


ansfyh                                                                    =
YYv*v*sqrt((v*v)+(u*u))+YYr*r*sqrt((u*u)+(v*v))+YYvv*v*abs_v+YYvr*v*abs_r
+YYrr*r*abs_r;
```

```
return ansfyh;
}


//calculation of sway rudder forces fyr //

double fyr (double u,double v,double r,double delta)
{
double ansfyr;

double po=1025.0;

double xPton=xP/sL;

double xRton=xR0/sL;

double rton=r*sL/sqrt(u*u+v*v);

double b=-asin(v/sqrt(u*u+v*v));

double bRton=b-(2*xRton*rton);

double   C1=-1.173;

double  wP=wP0*exp(C1*pow((b-xPton*rton),2));

double   wR=wR0*(wP/wP0);

double  ak=0.6*(1-wP)/(1-wR);

double   an=D/HR;

double   s=1-u*(1-wP)/(aan*pitch);

double    gs=an*ak*(2-(2-ak)*s)*s/(pow((1-s),2));


double C2=1.0;

double uR=u*(1-wR)*sqrt(1.0+C2*gs);

double CP=1/sqrt(1+0.6*an*(2.0-1.4*s)*s/pow((1-s),2));

double CS0=0.5;

double C3=0.45;

double CS=CS0;
```

```
double abs_rton = rton;
if (abs_rton < 0)
abs_rton = -abs_rton;

if (abs_rton <= (CS0/C3))
CS = C3*abs_rton;
double gama=CP*CS;

double vR=uR*gama*bRton;

double delta0=0.0;

// calculation of the delta0 for single srew ships//



double uP0=u*(1-wP0);

if ( (uP0/(aan*pitch)) > 0.2)

delta0=-3*(1-uP0/(aan*pitch))*(3.14/180.0) ;

if ( (uP0/(aan*pitch)) > 0.0 && (uP0/(aan*pitch)) < 0.2)

delta0=-12*(uP0/(aan*pitch))*(3.14/180.0);
else
delta0 = 0.0;

double  aR=delta+delta0-atan(vR/uR);

double
FIN=0.5*po*RAREA*(uR*uR+vR*vR)*((6.13*alamda)/(alamda+2.25))*sin(aR);

double aH0=0.32 ;

double uP=u*(1-wP);

double AJP=uP/(aan*pitch);

double aH=aH0*AJP/0.3;

if (AJP > 0.3)
aH = aH0;
ansfyr = -(1+aH)*FIN*cos(delta);

return ansfyr;

}
```

```
// calculation of the total sway forces on ship F2 //

double F2 (double u,double v,double r,double p,double phi,double
psi,double xi,double delta)
{

double ansF2 ;

ansF2 = -sm*r*u + fyh(u,v,r) + fyp(u) + fyr(u,v,r,delta);

return ansF2;
}

// CALCULATION OF YAW MOMENT OF SHIP //

//calculation of the yaw hull moment fnh //

double fnh (double u,double v,double r,double phi)
{
double ansfnh;

double abs_phi = phi;

double abs_r = r;

if (abs_phi < 0)
abs_phi = -abs_phi;
if (abs_r < 0)
abs_r = -abs_r;


ansfnh
=YNr*r*sqrt(u*u+v*v)+YNv*v*sqrt(u*u+v*v)+YNrr*r*abs_r+YNrrv*r*r*v/sqrt
(u*u+v*v)+YNvvr*v*v*r/sqrt(u*u+v*v)
+    YNphi*phi*(u*u+v*v)+YNvphi*v*abs_phi*sqrt(u*u+v*v)
+    YNrphi*r*abs_phi*sqrt(u*u+v*v);

return ansfnh;

}

//calculation of the yaw propeller moment fnp //

double fnp (double u)
{
double ansfnp;

int po =1025;
```

```
double NPC;


if (u>0 && aan <0)

NPC = 0.1;
else
NPC = 0.0;

ansfnp=po*(aan*aan)*(D*D*D*D*D)*NPC;

return ansfnp;

}

//calculation of the yaw rudder moment fnr//

double fnr (double u,double v,double r,double delta)
{
double ansfnr;

double xR=xR0;

ansfnr =(fyr(u,v,r,delta))*xR;

return ansfnr;

}

//calculation of the total yaw moment of ship F3 //

double F3 (double u,double v,double r,double p,double phi,double
psi,double xi,double delta)
{
double ansF3 ;

ansF3 = -sm*xG*r*u+fnh(u,v,r,phi)+fnp(u)+fnr(u,v,r,delta);

return ansF3;
}

// CALCULATION OF ROLL MOMENT OF SHIP //

//caclulation of the hull roll moment fkh //
double fkh(double u,double v,double r,double p,double phi,double xi)
{
double phiv=1.04712;
```

```
double dGM=0.2;

double XL=xi/WL;

double a=1/(pow(phiv,2));

double b=0.125;


return (double) (YKphi*(1+dGM*cos(6.28*XL))*((1-b*cos(6.28*XL))*phi -
a*(pow((1-b*cos(6.28*XL)),3))*pow(phi,3))
+YKp*p-zY*fyh(u,v,r));
}

//calculation of propeller roll moment fkp //

double fkp(double u)
{
double ansfkp;

ansfkp = 0;

return ansfkp;
}

//calculation of the rudder roll moment fkr //
double fkr (double u,double v,double r,double delta)
{
double ansfkr;

double zR=zR0;

ansfkr = -zR*fyr(u,v,r,delta);

return ansfkr;
}

//calculation of total roll moment of ship F4 //

double F4 (double u,double v,double r,double p,double phi,double
psi,double xi,double delta)
{
double ansF4;

ansF4=sm*zG*r*u+fkh(u,v,r,p,phi,xi)+fkp(u)+fkr(u,v,r,delta);

return ansF4;
}
```

```
//CALCULATION OF SURGE FORCES OF SHIP//

//calculation of the surge forces of the hull fxh //

double fxh(double u,double v,double r)
{
double ansfxh;

double Res=a1*u+a2*u*u+a3*u*u*u;

int IC;

if (u < 0.0)
IC = -1;
else
IC = 1;
ansfxh=-YYvdot*v*r-IC*YYrdot*r*r+YXvr*v*r-Res;

return ansfxh;
}

//calculation of the surge propeller force  fxp //

double fxp (double u,double v,double r)
{
double ansfxp;
double C1=-1.173;

double b=-asin(v/sqrt(u*u+v*v));

double xPton=xP/sL;

double rton=r*sL/sqrt(u*u+v*v);

double wP=wP0*exp(C1*pow((b-xPton*rton),2));


double tP=tP0*exp(C1*pow((b-xPton*rton),2));


double po=1025.0;

double AJ=u*(1-wP)/(aan*D);

double AKT=cc1+c2*AJ+c3*AJ*AJ;

double con=(1-tP)*po*D*D;
```

```
ansfxp=con*(aan*aan)*(D*D)*AKT;

return ansfxp;
}

//calculation of the rudder surge forces fxr //

double fxr (double u,double v,double r,double delta)
{
double ansfxr;
//aH0=-1.5*(0.48*D/HR)+1.0//
//C1=-4.0//

double aH0=0.32;

double C1=-1.173;

double b=-asin(v/sqrt(u*u+v*v));

double xPton=xP/sL;


double rton=r*sL/sqrt(u*u+v*v);
double wP=wP0*exp(C1*pow((b-xPton*rton),2));

double tP=tP0*exp(C1*pow((b-xPton*rton),2));

double tR=tP;

double uP=u*(1-wP);

double AJP=uP/(aan*pitch);

double aH=aH0*AJP/0.3;

if (AJP > 0.3)
aH=aH0;
ansfxr=tan(delta)*(1-tR)*fyr(u,v,r,delta)/(1+aH);
return ansfxr;
}

//calculation of the total surge forces of ship F1 //

double F1 (double u,double v,double r,double p,double phi,double
psi,double xi,double delta)
{
double ansF1;

ansF1 = sm*r*v+sm*xG*r*r-sm*zG*p*r+fxh(u,v,r)+fxp(u,v,r)
```

```
+   fxr(u,v,r,delta);

return ansF1;
}




void fcn (int neq,double t,double y[], double yprime[])
{

//      if (y[8]>0.62)
//      y[8]=0.62;
//if(y[8]<-0.62)
//      y[8]=-0.62;


double        tr=3.0, aw=3.0,bw=1.0 ,head=1.0;

double f1 = F1(y[0],y[1],y[2],y[3],y[4],y[5],y[6],y[8]);
double f2 = F2(y[0],y[1],y[2],y[3],y[4],y[5],y[6],y[8]);
double f3 = F3(y[0],y[1],y[2],y[3],y[4],y[5],y[6],y[8]);
double f4 = F4(y[0],y[1],y[2],y[3],y[4],y[5],y[6],y[8]);


yprime[0] = (1.0/(sm-YXudot))*f1;
yprime[1] = (1.0/DET)*(f2*BA1-f3*BA2+f4*BA3);
yprime[2] = (1.0/DET)*(-f2*BB1+f3*BB2-f4*BB3);
yprime[3] = (1.0/DET)*(f2*BC1-f3*BC2+f4*BC3);
yprime[4] = y[3];
yprime[5] = y[2];
yprime[6] = y[0]*cos(y[5])-y[1]*sin(y[5]);
yprime[7] =y[0]*sin(y[5])+y[1]*cos(y[5]);
//yprime[8] = tr*(-y[8]-aw*(y[5]-head)-aw*bw*y[2]); // use it for autopilot
else //
yprime[8] = 0;
}
```

Αρχειο Simulation.cpp

```
#include <math.h>
#include <imsl.h>
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <Inventor/Win/SoWin.h>
#include <Inventor/Win/viewers/SoWinExaminerViewer.h>
#include <Inventor/nodes/SoSeparator.h>
#include <Inventor/engines/SoTimeCounter.h>
#include <Inventor/fields/SoFieldContainer.h>
#include <Inventor/nodes/SoTransform.h>
#include <Inventor/engines/SoCompose.h>
#include <Inventor/sensors/SoTimerSensor.h>
#include <Inventor/sensors/SoSensor.h>
#include <Inventor/events/SoKeyboardEvent.h>
#include <Inventor/nodes/SoRotation.h>
#include <Inventor/nodes/SoEventCallback.h>
#include <Inventor/nodes/SoDirectionalLight.h>
#include <Inventor/VRMLnodes/SoVRMLBackground.h>



void fcn (int neq,double t,double y[],double yprime[]);
double F1(double,double,double,double,double,double,double,double);
double F2(double,double,double,double,double,double,double,double);
double F3(double,double,double,double,double,double,double,double);
double F4(double,double,double,double,double,double,double,double);


char *state;

ofstream output("simoutput.txt", ios::out);

int neq = 9;
double t=0.0;
double tend ;
double y[9]={0.1,0,0,0,0,0,0,0,0}; //initial values//
double step=0.0;
double aan;
```

```
static void
sensorcb (void *data, SoSensor *) {

step=step+0.1;
tend=step;
imsl_d_ode_runge_kutta (neq, &t, tend, y,state, fcn);
output<<step<<setw(15)<<y[0]<<setw(15)<<y[1]<<setw(15)<<y[2]<<setw(1
5)<<y[3]<<setw(15)<<y[4]<<setw(15)<<y[5]<<setw(15)<<y[6]<<setw(15)<<y[
7]<<setw(15)<<y[8]<<endl;


SoTransform *mycast = (SoTransform*)data;
mycast->translation.setValue(y[6],y[7],0);
SbRotation *rotx = new SbRotation;
rotx->setValue(SbVec3f(1,0,0),-y[4]);
SbRotation *rotz = new SbRotation;
rotz->setValue(SbVec3f(0,0,1),y[5]);
SbRotation *rot = new SbRotation;
*rot=(*rotx)*(*rotz);
mycast->rotation.setValue(*rot);

}

void ruddercb (void*,SoEventCallback *eventCB){


const SoEvent *event = eventCB->getEvent();
if (SO_KEY_PRESS_EVENT(event,F)) {
y[8]=y[8]+0.1;
cerr <<"Rudder deflection = "<<y[8]<<setw(15)<<"rad"<<endl;

eventCB->setHandled();
}
else if (SO_KEY_PRESS_EVENT(event,H)) {
y[8]=y[8]-0.1;
cerr <<"Rudder deflection = "<<y[8]<<setw(15)<<"rad"<<endl;
eventCB->setHandled();

}
}
void rpmcb (void*,SoEventCallback *eventCB){



const SoEvent *event = eventCB->getEvent();
if (SO_KEY_PRESS_EVENT(event,T)) {
aan=aan+1;
cerr <<"Propeller velocity = "<<aan<<setw(15)<<"rad/sec"<<endl;
```

```
eventCB->setHandled();
}
else if (SO_KEY_PRESS_EVENT(event,G)) {
aan=aan-1;
cerr <<"Propeller velocity = "<<aan<<setw(15)<<"rad/sec"<<endl;
eventCB->setHandled();

}
}

main(int, char ** argv)
{
int nstep;


double head = 1.0;

HWND window = SoWin::init(argv[0]);
if (window==NULL)
exit(1);
SoSeparator * root = new SoSeparator;
SoSeparator * roro = new SoSeparator;
SoSeparator *port = new  SoSeparator;
SoTransform *trans = new SoTransform;

imsl_d_ode_runge_kutta_mgr(IMSL_ODE_INITIALIZE,&state,IMSL_TOL,
0.0000001,IMSL_NSTEP,              &nstep,IMSL_MAX_NUMBER_STEPS,
50000000, 0);

SoWinExaminerViewer * viewer = new SoWinExaminerViewer(window);
SoTimerSensor *mysens = new  SoTimerSensor(sensorcb,trans);
mysens->setInterval(0.1);
mysens->schedule();
SoEventCallback *rudder = new  SoEventCallback ;

rudder->addEventCallback(SoKeyboardEvent::getClassTypeId(),ruddercb,NULL);

SoEventCallback *rpm = new  SoEventCallback ;

rpm->addEventCallback(SoKeyboardEvent::getClassTypeId(),rpmcb,NULL);


SoTransform *syrostrans = new SoTransform;
syrostrans->translation.setValue(0,0,6.2);


SoDirectionalLight *mylight = new SoDirectionalLight;
```

```
SoInput myinput;
if (!myinput.openFile("Alkyon.wrl"))
return (1);
SoSeparator *ship = SoDB::readAll(&myinput);
if (ship == NULL)
return (1);


SoInput portinput;
if (!portinput.openFile("syros.wrl"))
return (1);
SoSeparator *syros = SoDB::readAll(&portinput);
if (syros == NULL)
return (1);

roro->ref();

roro->addChild(rudder);
roro->addChild(rpm);
roro->addChild(trans);
roro->addChild(ship);

port->addChild(syrostrans);
port->addChild(syros);

root->ref();
root->addChild(mylight);
root->addChild(port);
root->addChild(roro);

viewer->setSceneGraph(root);
viewer->show();

SoWin::show(window);
SoWin::mainLoop();
delete viewer;
root->unref();
}
```

Όσον αφορά το αρχείο simulation.cpp η εισαγωγή του λιμανιού της Τήνου αντί του λιμανιού της Σύρου γίνεται τροποποιώντας την εντολή

```
SoInput portinput;
if (!portinput.openFile("syros.wrl"))
return (1);
```

που βρίσκεται προς το τέλος του κώδικα με εισαγωγή του αρχείου tinos.wrl ως εξής:

```
SoInput portinput;
if (!portinput.openFile("tinos.wrl"))
return (1);
```